

CHAPTER S

SAVE MEMORY ON TAPE

S-0. This chapter describes the @SAVE command and its counterpart in assembly program. The purpose of @SAVE is to copy the content of a block of the volatile RAM onto the non-volatile tape so that it can be retrieved later. The retrieving is called "Load" and is discussed in Chapter L.

S-1. Save Machine Language Programs

Machine Language programs (or a block of memory of arbitrary content) can be copied on tape with the BASIC command:

@ (#d)SAVEN,addr,lnth(,aust)

where d is the optional drive number, n is the file number, addr is the address of the block of memory, lnth is the length of the block, and aust is the optional auto-start address. All these parameters can be BASIC expressions with the limitations:

d should be between 0 and 7
n should be between 1 and 99
addr,.lnth,aust should be between* -32768 and 32767

*NOTE: In ESF firmware version 3.2, these three parameters cannot be expressions, and must be numbers between 0 and 65535. In version 4.1 these parameters can be either numbers or BASIC expressions. However, due to the way BASIC handles integers, the range of value is between -32768 and 32767. Hex 0000 through 7FFF are integers 0 through 32767. Hex 8000 through FFFF are integers -32768 through -1.

The optional parameter aust is recorded on tape and used by @LOAD to determine what to do after the file is read back from tape and loaded into memory. If this option is not specified, the firmware will supply the default value 12309 (Hex 3015) which causes the firmware to return control to BASIC. If the machine language program is a stand-alone program (as opposed to a subroutine or patch to BASIC or other programs), it is recommended that the execution starting address should be used as aust.*

*NOTE: The ESF firmware version 3.2 has no means to override the autostart, and thus, much stand-alone software is shipped with the default aust 12309 so that the user can make copies. This is no longer necessary with version 4.1. If the user wants to override autostart to make copies, he (or she) should

press and hold the shift key while loading the program. The firmware will not start to execute the loaded program, but display the address, length, and autostart on the screen on completion of loading.

The assembly language counterpart of the above BASIC command is:

```
LD      HL,(40B1H)
INC     HL                      ; Optional, set drive #
LD      (HL),0F0H+drive #
LD      A, file #
LD      HL, address of block
LD      BC, length of block
LD      DE, autostart*
CALL    300CH
```

*NOTE: The autostart must be supplied and is not an optional parameter. 3015H can be used if control should be returned to BASIC after loading. An autostart of 0000 should be used only if the block to be saved is a BASIC program.

S-2. Restrictions and Other Comments

In principle, the SAVE command or assembly program in the last section can be used to save any part of the memory, including the ROM and the video refresh RAM. However, the following should be noted:

- (a) When used to save the video refresh RAM (Hex 3C00 to 3FFF), the unwanted message: "WRITING.." will also be recorded. Furthermore, after the loading, the message: "DONE" will also show up on the screen. One way to avoid this is to disable the display output routine before the SAVE and LOAD command, and re-enable it afterwards. For example:

```
100      REM compose a beautiful picture
        .
        .
        .
600      POKE 16414,87 : REM disable display output
610      @SAVE3,15360,1024
620      POKE 16414,88 : REM enable display output
700      REM do other things
        .
        .
        .
```

```
900      POKE 16414,87
910      @LOAD3
920      POKE 16414,88
```

This method is somewhat unsafe. If an error occurred while the display output is disabled, the error message will not be displayed, and the computer will appear to be dead. One can recover from this by typing "(BREAK)" and "POKE 16414,88 (Enter)" blindly.

- (b) The RAM location Hex 401A is used by the ESF firmware (version 4.1). If this location is included in the block to be saved, a "verify error" will occur during SAVE. Subsequently, a "checksum error" will occur during LOAD. There is no harm done in either case.
- (c) If the stack is included in the block to be saved, a "verify error" will occur during SAVE. No harm is done. If the stack at load time is included in the block to be loaded, the result is usually catastrophic. The stack at load time is independent of the stack at save time, and is not directly under the control of the programmer. For BASIC, the top of stack is usually about 50 bytes below the top of available RAM and is pointed by the contents of location Hex 40E8-40E9. After SYSTEM, the top of stack is usually at Hex 4288.

S-3. Save BASIC Programs

The BASIC command to save the current BASIC program is:

@ (#d)SAVE n

where d is the optional drive number, which can be an expression with value between 0 and 7, and n is the file number, which can be an expression with value between 1 and 99. This command will cause the firmware to find the block of memory, which contains the current BASIC program, save this block with an autostart of 0000. Assembly program can use the same code as in section S-1 to save a BASIC program, but the address and the length of the block must be given. The beginning address of the BASIC program is usually stored in the location Hex 40A4-40A5. The end of BASIC program is marked by three consecutive bytes of zeros. A subroutine at

Hex 1AF8 can be used to find this marker.

```

CALL    1AF8H           ;find end of BASIC
INC     HL              ;HL -] end + 1
LD      DE,(40A4H)      ;DE -] begin of BASIC
SUB     A               ;clear carry-borrow
SBC     HL,DE           ;HL = length of BASIC
EX      DE,HL           ;HL -] begin
PUSH    DE
POP      BC              ;BC = length
LD      DE,0            ;autostart must be 0
LD      A,file #
CALL    300CH           ;SAVE

```

The autostart = 0 is a special flag for "LOAD" to relocate BASIC program. See Chapter L for details.

S-4. Sequence of Events During SAVE

The files on tape are separated and identified by file marks. The @NEW command will write an "End of file 0" at the beginning of the tape. The @SAVE1 command will execute the following sequence:

- (a) Start the motor and search for "End of file 0".
- (b) When found, start to write the block on tape.
- (c) When finished with the block, keep motor running to leave a gap on tape.
- (d) After the gap, write "End of file 1" mark.
- (e) Search for "End of file 0" again.
- (f) When found, start to verify the block on tape with what is in memory.
- (g) When finished with the block, stop the motor. Because of the gap in step (c), the tape will stop before the "End of file 1" reaches the head.

The @SAVE2 command will be executed similarly, except all the file numbers are one bigger now.

S-5. Speedy SAVE

Using the firmware, it takes at least two loops to finish the @NEW, and at least one loop to @SAVE each file. This can be very time consuming to copy a long tape with many files. The

following sequence can be used to save files without verify:

(a) Instruct the operator to insert a blank tape into the drive.

(b) Execute the following to find the beginning of tape:
CALL 3000H

(c) Ready the file in memory to be saved. (See Section L-5.)

(d) Execute the following to write "End of file-1" and then the block:

```

LD      A,(file #)      ;file
DEC     A               ;file-1
CALL    3021H           ;write "End of file-1"
JR      NZ,FIN
CALL    3733H           ;motor on again
LD      A,(file #)
LD      L,A
CALL    Z,3592H         ;start the block
JR      NZ,FIN
LD      HL,(addr)
CALL    363BH           ;write the block addr
JR      NZ,FIN
PUSH    AF              ;wait a while
LD      A,(IX)
POP     AF
LD      HL,(autostart)
CALL    363BH           ;write the autostart
JR      NZ,FIN
RET     NZ
LD      IX,memory       ;actual addr of block
LD      HL,(length)
CALL    3567H           ;write the block
FIN:    NOP              ;finished, ck error if non-0

```

(e) Repeat step (c) and (d) for all files to be saved.

(f) Execute the following to write the last "End of file":

```

LD      A,(file #)
CALL    3021H

```

(g) Inform the operator that the tape is finished.

A program called "COPYCAT" uses the method outlined above to enable user to write BASIC program to control the copy process. Effective use of "COPYCAT" calls for two or more drives in the system.

CHAPTER L

LOAD MEMORY FROM TAPE

L-0. This chapter is an attempt to explain the actions of the @LOAD command. These actions are controlled by the parameters given during the @SAVE, which is described in chapter S, and also by the conditions during LOAD.

L-1. Loading Machine Language Programs

The so-called machine language program can actually be any memory dump. This kind of file is identified by a non-zero autostart. The action of @LOAD is rather simple. The file is read into the memory as specified during @SAVE, no check is made on where it is going to be loaded or how long the block is. If this block covers the video refresh RAM, the screen will show it. If this block overlays the stack, the system will crash. After the block is read, (and if the system did not crash), the firmware will check to see if the "Shift" key is depressed. If so, the block address, length, and the autostart will be displayed on the screen, and the FD error code is passed back to BASIC. If the "Shift" key is not depressed, the firmware will jump to the autostart address. Included in the firmware is the routine at Hex 3015, which will restore the registers and make a smooth return to the BASIC interpreter. Thus, SAVED files with autostart set to the default Hex 3015 will not disrupt the normal execution of BASIC program with an @LOAD embedded in the code.

L-2. Loading BASIC Programs by Direct Command

A BASIC program saved on tape is identified by the zero autostart. The action of @LOAD does not depend on whether the "Shift" key is depressed but does depend on whether the @LOAD command is a direct command typed on the keyboard or not. If it is a direct command:

- (a) The block length is checked against the total available memory space calculated from:

```
LD      HL,(40A0H)      ;string space
LD      DE,(40A4H)      ;BASIC begin
SBC     HL,DE
LD      DE,100          ;stack space
SBC     HL,DE
```

where (40A0H) is set by the CLEAR n command to be:

(40B1H)-n with default value of n = 50, and (40B1H) in turn is usually set by the "MEMORY SIZE?" at power up, or subsequently changed by the ESF firmware or other programs.

- (b) If there is enough memory, the tape is loaded into memory pointer by the content of locations Hex 40A4-40A5, and not by the parameters recorded during @SAVE.
- (c) After it is loaded, a special subroutine is called to clean up the links inside the loaded BASIC program. This is necessary because the contents of Hex 40A4-40A5 during @SAVE and @LOAD may be different. As a consequence, the BASIC program is loaded into a different block of memory and the links inside are all wrong.
- (d) All the variables and strings are cleared. Control is back to BASIC direct mode.

L-3. Loading BASIC Program by Another BASIC Program

When the @LOAD is executed as part of a BASIC program, the actions taken by the ESF firmware is quite different. It is designed to be able to do program overlay or chaining.

- (a) The block length is checked against the available memory space exclusive of the variable already defined:


```
LD      HL,(40F9H)      ;variable space
LD      DE,(40A4H)      ;BASIC begin
SBC     HL,DE
```

where (40F9H) is set by the CLEAR command to point one above the end of the current BASIC program. Thus, in order to have enough space to load, the current program must be larger than or equal to the one being loaded. See Section L-4 for methods to overcome this restriction.

- (b) and (c) Same as in Section L-2.
- (d) All the variables and strings are preserved. Control is back to BASIC to start executing the first statement of the program just loaded.

L-4. Program Overlay

When a program is too big to fit into the available memory, it is often possible to divide the big program into logically operable small segments and load only one segment into memory at a time. The special actions of @LOAD described in Section L-3 make this possible to do in the TRS-80 system.

For example, for some unknown reason, one decides to write an Editor-Assembler-Debugger in BASIC. It is clear that during editing, the code for Assemble and Debug need not be in memory, and vice versa. Thus, one can divide this into three segments. In the Editor segment, you can type the command "A" and the program will do an @LOAD2 which will wipe out the Editor segment and load in and start the Assembler. Or, you can type the command "Z", and the editor will do an @LOAD3 which will also wipe out the Editor and load in and start the Debugger. While in the Debugger, the command "E" will cause an @LOAD1 and load back the Editor. One important point is that during all this loading and reloading, the variables and strings - which contain the source and object code you are trying to edit, assemble, and debug - should always be preserved and available to the different segments of the program. Another perhaps less important thing is the new segment being read in should start automatically. The @LOAD does precisely these.

The segmentation of the above example is obvious. But even less obvious problems can always be divided and conquered. The few problems with overlay on TRS-80 are:

- (a) The segment that is loaded first by direct command @LOAD has to be the largest segment. Otherwise, there will be a OM error when it tries to @LOAD a larger segment. One way to overcome this is to put two POKE's in the first segment:

```
10      POKE 16633, low-order
20      POKE 16634, high-order
```

where the low-order, high-order are the contents of location 16633, 16634 respectively when the largest segment is @LOAD by direct command.

- (b) Some of the strings that were defined may appear to be wrong after overlay. This can be avoided if one does not use assignments like:

```
10      A$="this one"
```

but use instead:

```
10      A$="this one"+""
```


- (c) READ data commands get lost or cause an OD error. The solution is to put a RESTORE command at the beginning of that segment.

L-5. Loading Programs by Assembly Program

The firmware does not support assembly programs to load programs. But this can be done and is used in the "COPYCAT" program described in Section S-5.

```

                LD      HL,max-block size
                PUSH    HL
                LD      H,file #
LD2:            CALL    3734H          ;motor on
LD3:            CALL    3649H          ;find sync
                JR      NZ,LD5
                LD      A,D
                OR      A
                JR      LD3            ;data file
                JP      M,LD3          ;end of file
                SUB     H              ;found?
                NOP
                JR      NZ,LD3          ;not yet
                CALL    370BH          ;found it
LD5:            JP      NZ,ERROR
                LD      H,D
                LD      (addr),HL      ;save addr
                LD      IX,memory      ;actual addr
                PUSH    HL
                INC     HL
                POP     HL
                CALL    370BH          ;read 2 bytes
                JR      NZ,LD5
                LD      H,D
                LD      (autostart),HL ;save autostart
                LD      IY,LD1         ;fake call
                JP      31B8H
LD1:            LD      (length),DE    ;save length

```

CHAPTER D

DATA I/O

D-0. This chapter discusses the steps needed in assembly programs to output or input a data file. The Data I/O program for BASIC and a patch for Radio Shack's EDTASM are used as examples.

D-1. Output a Data File

(a) To open the file:

- (i) Define a buffer, and indicate that the buffer is empty.
- (ii) Find the "End of file-1":
 - LD A,file #
 - CALL 300FH

Note that this will start the drive motor and can take up to one full loop to find the beginning of the file. If the user knows that the position of the tape is correct, step (ii) can be omitted and thus save a considerable amount of time.

(b) To output a byte:

- (i) Put the byte in the buffer, and move the buffer pointer by one.
- (ii) If buffer full, write it on tape:
 - LD HL,addr of buffer
 - LD BC,length of buffer
 - CALL 3006Hindicate that the buffer is empty again.
- (iii) Return.

(c) To close the file:

- (i) Write buffer if not empty, and write end of file:
 - LD HL,addr of buffer
 - LD BC,# of bytes in buffer
 - LD A,file #
 - CALL 3027H
- (ii) Free the buffer.

D-2. Input a Data File

(a) To open the file:

(i) Define a buffer and indicate that it is empty.

(ii) Find the "End of file n-1"

```

LD      A,file #
CALL    300FH

```

(b) To get a byte:

(i) If buffer empty, read a record:

```

LD      HL,addr of buffer
LD      BC,length of buffer
CALL    3003H
LD      (# of bytes in buffer),BC

```

(ii) Get the byte from buffer.

(iii) Decrease the # of bytes in buffer by one.

(c) To close the file:

Free the buffer.

D-3. The Data I/O Program for BASIC

The Data I/O program for BASIC supplied with every ESF follows the procedure described in Sections D-1 and D-2. A listing of the program can be found in Appendix. This program is actually loaded into location 6C00H and relocated. All of the addresses in the listing marked with a single quote ' are changed to reflect the actual relocated addresses.

Note that this is a buffered I/O and is quite different from the unbuffered TRS-80 cassette Data I/O. The following two programs will write identical tape files.

Program 1:

```

10      @OPEN1
20      FOR I=1 TO 10
30      @PRINT I
40      FOR J=1 TO 10000
50      NEXT J
60      NEXT I
70      @CLOSE

```

Program 2:

```

10      @OPEN1
20      @PRINT 1,2,3,4,5,6,7,8,9,10
30      @CLOSE

```

Equivalent programs on TRS-80 cassette Data I/O will produce a much longer tape in case 1 as compared with case 2.

D-4. Patch to Radio Shack's "EDTASM"

A patch to Radio Shack's "EDTASM" to use ESF for source storage is shown on the next page as another example.

```

00100 :           ESF PATCHES TO "EDTASH"
00200 :           -----
00300 ; PART # 1: KEYBOARD DEBOUNCE
4301      00400      ORG      4301H
4301 5C37    00500      DEFW      375CH
00600 ;
00700 ; PART # 2: PROTECT HIGH MEMORY
4695      00800      ORG      4695H
4695 2AB140  00900      LD        HL,(40B1H)
4698 1805    01000      JR        469FH          ;FIND BOF
01100 ;
01200 ; PART # 3: WRITE SOURCE ON ESF
4D23      01300      ORG      4D23H
4D23 3A2841  01400 WRITE LD      A,(4128H)      ;GET NEXT CH AFTER "W"
4D26 D630    01500      SUB      '0'          ;CONVERT IT TO FILE #
4D28 F5      01600      PUSH     AF
4D29 CD0F30  01700      CALL     300FH
4D2C 2012    01800      JR        NZ,ERROR
4D2E 0112A3  01900      LD        BC,-5CEEH      ;-(BEGIN ADDR. OF SOURCE)+2
4D31 2A1541  02000      LD        HL,(4115H)      ;(END ADDR. OF SOURCE)
4D34 09      02100      ADD      HL,BC
4D35 E5      02200      PUSH     HL
4D36 C1      02300      POP      BC              ;BC = LENGTH OF SOURCE
4D37 21F05C  02400      LD        HL,5CF0H        ;(BEGIN ADDR. OF SOURCE)
4D3A F1      02500      POP      AF              ;FILE # AGAIN
4D3B CD2730  02600      CALL     3027H          ;WRITE FILE + EOF
4D3E C3      02700      RET      Z
4D3F E5      02800      PUSH     HL
4D40 E1      02900 ERROR POP      HL              ;IN CASE OF AN ERROR
4D41 C630    03000      ADD      A,'0'          ;CONVER ERROR CODE TO ASCII
4D43 E67F    03100      AND      7FH
4D45 21AE48  03200      LD        HL,48AEH        ;POINT TO MESSAGE
4D48 C33147  03300      JP        4731H          ;GO PRINT AND RECOVER
03400 ;
03500 ; PART # 4: READ SOURCE FROM ESF
4D4B 3A2841  03600 READ LD      A,(4128H)      ;GET CH AFTER "L"
4D4E D630    03700      SUB      '0'          ;CONVERT ASCII TO FILE #
4D50 CD0F30  03800      CALL     300FH          ;FIND THE BOF
4D53 20EC    03900      JR        NZ,ERROR+1
4D55 0112A3  04000      LD        BC,-5CEEH      ;-(BEGIN ADDR. OF SOURCE)+2
4D58 2A1341  04100      LD        HL,(4113H)      ;(MAX MEMORY ADDR.)
4D5B 09      04200      ADD      HL,BC
4D5C E5      04300      PUSH     HL
4D5D C1      04400      POP      BC              ;MAX LENGTH OF SOURCE

4D5E 21F05C  04500      LD        HL,5CF0H        ;(BEGIN ADDR. FOR SOURCE)
4D61 221141  04600      LD        (4111H),HL
4D64 CD0330  04700      CALL     3003H          ;READ FILE
4D67 20D8    04800      JR        NZ,ERROR+1
4D69 09      04900      ADD      HL,BC
4D6A 2B      05000      DEC      HL
4D6B 2B      05100      DEC      HL
4D6C 221541  05200      LD        (4115H),HL      ;-(END ADDR. OF SOURCE)
4D6F C9      05300      RET
468A      05400      END      468AH
00000 TOTAL ERRORS

```

CHAPTER T

TAPE FORMAT

T-0. This chapter deals with the tape format used in the ESF for TRS-80 and some of the inner details of the low level subroutines in the firmware. One does not need to know these details in order to use the ESF. One might even get more confused by reading it. This chapter is for the curious and the desperate. The curious should read it without taking it too seriously, and never try to use the subroutines named in it. The desperate should use this chapter as a guide, read the firmware listing carefully, worry about all the registers, count the machine cycles (hereafter referred to as TRS-80 cycles) between input or output instructions, etc. If after all these, the desperation turns into frustration, the author of the firmware (and this manual) expresses his deep sympathy.

T-1. Recording Method

The so-called frequency modulation (FM) encoding is used in the ESF for TRS-80. This is the same method that is used in most single density floppy disks. Data (which also includes images of programs in this context) are written on the magnetic media one bit at a time in blocks called records or sectors. A magnetic flux change is written at the leading edge of each bit cell. An additional flux change is written at the center of the bit cell if, and only if, the bit is a "1". For the ESF, each bit cell is 150 micro-seconds, or 248 TRS-80 cycles.

The subroutine "WRBIT" in the ESF firmware will write n consecutive bits assuming that the I/O port number is in C register, the number n is in B register, and the bits to be written are in D register. This subroutine will also check the write-protect detector and the end-of-tape (EOT) detector. If either one is set, it returns a 01 or a 04 respectively in the A register, and an NZ in the F register. Otherwise, Z will be returned in the F register.

"WRBIT" reverses the write-current every 248 TRS-80 cycles for each bit of data. (These are the "clocks".) It also reverses the current at 124 TRS-80 cycles if the particular bit is a "1". (These are the "data".) "WRBIT" takes 75 TRS-80 cycles before writing the first "clock" and 28 TRS-80 cycles after last "data" to return to the caller. The caller should use up 21 TRS-80 cycles between calls to meet the 124 cycle requirement. Note that RAM location 401AH is used by "WRBIT".

The "DJNZ RB2" loop of the subroutine "RDBYT" is responsible to read the bits back from tape. It loops either in the "JP P,RB4" loop, or the "JP M,RB5" loop for a negative going or a positive going "clock" respectively. Then it waits 175 TRS-80 cycles and reads the polarity of the magnetic flux. If the polarity has changed, then the bit is a "1", else it is a "0". This bit is saved in the D register. (See next section for a full description of "RDBYT".) The 175 cycle wait is optimal for speed tolerance and other considerations to recover data with 248 cycle bit cell.

T-2. Byte Format

The 8 bits of a byte is written consecutively starting at the least significant bit (bit 7). In addition, a parity bit is written after the most significant bit (bit 0). The parity bit is "1" if the total number of 1's in the byte is even. The parity bit is "0" if the total number of 1's in the byte is odd. This is called "odd parity", but ironically, "odd" parity makes the total number of flux changes "even" in FM recording. An even number of flux changes means the polarity of the magnetic flux ends up the same as it started with. This makes the parity bit very easy to generate or to check.

The subroutine "WRBYT" in the ESF firmware will write the byte in the D register on tape. The parity bit is generated by forcing the polarity of the write current to be positive at the center of the parity bit cell. "WRBYT" will detect write-protect and EOT as it calls "WRBIT". "WRBYT" also adds the contents of D to register E for checksum.

The subroutine "RDBYT" will read a byte from the tape and put it in the D register. The previous contents of D is added to RAM location 401AH for the checksum. "RDBYT" does not check the parity of the current byte, but does check that of the previous byte simply by looking at the polarity of the magnetic flux. "RDBYT" also scans the BREAK key. It returns a 08 or a 01 in the A register if parity error or BREAK respectively and with an NZ in the F register. Normal return has 00 in A and Z in F.

T-3. Record Format

As stated before, bits and bytes are never written alone, but always in blocks called a record. A record starts with 512

bits of 0's followed by a single bit of "1", and a "sync byte". After the sync byte are the "record type" and the type dependent bytes. The stream of "0" bits helps the read routine to find the beginning of a record. Note that:

(a) Since 0's have "clocks" only, the read routine cannot mistake "data" as "clock" or vice versa. Thus, this makes it easy to achieve the so-called "bit synchronization".

(b) Since data bytes have odd parity, there can never be more than 16 consecutive 0's in the data stream. Thus, the long stream of 0's is unique and un-ambiguous.

The single bit of "1" after the long stream of 0's mark the byte boundary and helps the read routine to achieve "byte synchronization". The sync byte is used to double-check the byte synchronization and is redundant.

In the ESF firmware, the subroutine "WPREAM" writes the above stream of bits plus the record type byte (which will be defined in the next three sections). The record type byte is passed to this subroutine in the L register, and the subroutine also clears the E register for checksum computation.

The subroutine "RPREAM" searches for the beginning of record stream and reads the record type byte into D register. It also clears RAM location 401AH for checksum computation.

T-4. File Mark Record

Records on an ESF tape are divided into groups called files. A special record called file mark record is used to separate the files. A file mark record has a record-type-byte of between hex 80 and FF, and two arbitrary bytes following the record type. (The extra two bytes enable the read routine to check the parity of the record type byte.)

An empty tape (with 0 files) should have a file mark record with hex FF near the beginning of the tape. A tape with 1 file should have the above file mark record FF followed by one or more data records (see Section T-5) or program records (see Section T-6), followed by another file mark record FE. In general, file n consists of one/more data/program records preceded by a file mark record with the 1's complement of n as record type, and followed by a file mark record with the 2's complement of n as record type. File number n must be consecutive and started with 1. There cannot be more than 127 files.

The subroutine "WRPRE" is used to write file mark record. The record type is set up by the caller and stored in the L register.

T-5. Data Record

A data record has a record-type-byte of 00. This is followed by a two-byte data size count, the body of data, and a checksum byte and two arbitrary bytes. The data size count is the byte count of the body of data, and is written lower-byte first, high order byte next. The body of data is usually a dump of memory block. The checksum byte is the two's complement of the sum of all the bytes before it in this record. The two arbitrary bytes following checksum enable the read routine to check the parity of the checksum.

The subroutine "WRTWO" and "RDTWO" are used to write and read the data size count. The subroutines "WBLOCK" and "RBLOCK" are used to write and read the remaining part of a data record.

T-6. Program Record

A program record has a record-type-byte of 01 through 7F hex. This is followed by the sequence:

(a) A two-byte program address, low order first, high order next. This is the beginning address of the block of memory being copied to tape during "SAVE". It is also used as the beginning address of memory for "LOAD", if and only if, item (b) below is not 0000.

(b) A two-byte auto-start address, low order first, high order next. If this is not 0000, the ESF firmware will JUMP to this address after the program is loaded into the memory during a "LOAD". If this auto-start is 0000, special action is taken during "LOAD".

(c) A two-byte program size count, low order first, high order next.

(d) The body of the program.

(e) A checksum byte which is the two's complement of all the bytes in this record (including the sync byte) before itself.

(f) Two arbitrary bytes.

(000D)	0001	CR	EQU	00H
(0016)	0002	SYNC	EQU	16H
(00FB)	0003	CEI	EQU	0FBH
(401A)	0004	BSRAM	EQU	401AH
(2B02)	0005	BSINT	EQU	2B02H
(40A2)	0006	BSLIN	EQU	40A2H
(0FAF)	0007	BSFRN	EQU	0FAFH
(40B1)	0008	BSMEM	EQU	40B1H
(19A2)	0009	BSERR	EQU	19A2H
(1D1E)	0010	BSCONT	EQU	1D1EH
(1E4A)	0011	BSFCE	EQU	1E4AH
(1AF8)	0012	BSFIND	EQU	1AF8H
(40E6)	0013	BSHL	EQU	40E6H
(2B1C)	0014	BSIEXP	EQU	2B1CH
(4012)	0015	BSINTH	EQU	4012H
(4004)	0016	BSRST2	EQU	4004H
(032A)	0017	BSPRTC	EQU	032AH
(1E83)	0018	BSCLR	EQU	1E83H
(1A2B)	0019	BSRDY	EQU	1A2BH
(1D78)	0020	BSINC	EQU	1D78H
(1D5B)	0021	BSEXE	EQU	1D5BH
(40A4)	0022	BSBGNP	EQU	40A4H
(40F9)	0023	BSENDP	EQU	40F9H
(40A0)	0024	BSMAXP	EQU	40A0H
(1997)	0025	BSSYNE	EQU	1997H
(1AE8)	0026	BSFIX	EQU	1AE8H
(00AD)	0027	BSCSV	EQU	0ADH
(00A7)	0028	BSCLD	EQU	0A7H
(00BB)	0029	BSCNW	EQU	0BBH
(4016)	0030	BSKI	EQU	4016H
(4036)	0031	BSKS	EQU	4036H
(03FB)	0032	BSKR	EQU	03FBH
(0060)	0033	BSDLY	EQU	0060H
(3880)	0034	BSSFT	EQU	3880H
(3801)	0035	BSKEY	EQU	3801H
(3840)	0036	BSBRK	EQU	3840H
	0037	;		
	0038	*** **	VECTORS	*** **
	0039	;		
0000	0040		ORG	3000H
3000	034232	0041	JP	REWIND
3003	03DA32	0042	JP	READ
3006	035C33	0043	JP	WRITE
3009	036D33	0044	JP	WEOFX
300C	038233	0045	JP	SAVEA
300F	037F34	0046	JP	FBOF
3012	038734	0047	JP	SELECT
3015	2AE640	0048	BASIC	LD HL, (BSHL)
3018	031E1D	0049	BS1	JP BSCONT
301B	036534	0050		JP FBOFX
301E	032A33	0051		JP WRITEX
3021	036E33	0052		JP WEOF
3024	035F32	0053		JP NEWA
3027	036433	0054		JP WBOF
302A	039A34	0055		JP ERROR
302D	03D230	0056		JP GETN
3030	037930	R 0057		JP AUTO

```

0058 ;
0059 ;*** *** >SYSTEM *** ***
0060 ;      *? /1234N
0061 ;
0062 ;WHERE N=0 FOR @LOAD
0063 ;      1 FOR @LOAD 1
0064 ;      2 FOR @LOAD 2
0065 ;      3 FOR @LOAD 3
0066 ;      4 FOR @LOAD 4
0067 ;      5 FOR NO LOAD
0068 ;      6 FOR KEYBOUNCE
0069 ;
3033 00      0070      NOP
3034 3C      0071 START INC      A
3035 3C      0072      INC      A
3036 3C      0073      INC      A
3037 3C      0074      INC      A
3038 3C      0075      INC      A
3039 3C      0076      INC      A
303A 321A40  0077      LD      (BSRAM),A
303D 218230  0078      LD      HL,CHECK
3040 220440  0079      LD      (BSRST2),HL
3043 2AB140  0080      LD      HL,(BSMEM)
3046 3619    0081      LD      (HL),BSSYNE/256
3048 2B      0082      DEC      HL
3049 3697    0083      LD      (HL),BSSYNE MOD 256
304B 2B      0084      DEC      HL
304C 36C3    0085      LD      (HL),0C3H
304E 2B      0086      DEC      HL
304F 36F0    0087      LD      (HL),0F0H
3051 2B      0088      DEC      HL
3052 22B140  0089      LD      (BSMEM),HL
3055 21A234  0090      LD      HL,HEAD
3058 CD7935  0091      CALL    PRTSTG
305B 113200  0092      LD      DE,50
305E CD831E  0093      CALL    BSCLR
3061 3A1A40  0094      LD      A,(BSRAM)
3064 ED44    0095      NEG
3066 280E    0096      JR      Z,ST1
3068 215C37  0097      LD      HL,DEBNC
306B 221640  0098      LD      (BSKI),HL
306E C606    0099      ADD     A,6
3070 4F      0100      LD      C,A
3071 FE05    0101      CP      5
3073 C25531  0102      JP      NZ,LOADA
3076 C32B1A  0103 ST1    JP      BSRDY
3079 CDF81A  0104 AUTO  CALL    BSFIND
307C 2AA440  0105      LD      HL,(BSBGNP)
307F 2B      0106      DEC      HL
3080 1896    0107      JR      BS1
0108 ;
0109 ;*** MAKE BASIC LOOK FOR '@' ***
0110 ;
3082 E3      0111 CHECK  EX      (SP),HL
3083 7D      0112      LD      A,L
3084 FE5B    0113      CP      BSEXE MOD 256
3086 2003    0114      JR      NZ,CH1

```

3088	7C	0115	LD	A,H
3089	FE1D	0116	CP	BSEX/256
308B	E3	0117 CH1	EX	(SP),HL
308C	C2781D	0118	JP	NZ,BSINC
308F	D7	0119	RST	10H
3090	F5	0120	PUSH	AF
3091	E6DF	0121	AND	ODFH
3093	FE40	0122	CP	'@'
3095	2802	0123	JR	Z,CH3
3097	F1	0124	POP	AF
3098	C9	0125	RET	
3099	F1	0126 CH3	POP	AF
309A	F1	0127	POP	AF
309B	D7	0128	RST	10H
309C	2805	0129	JR	Z,CH4
309E	FE23	0130	CP	'#'
30A0	2016	0131	JR	NZ,CH5
30A2	D7	0132	RST	10H
30A3	2863	0133 CH4	JR	Z,SAO
30A5	CD1C2B	0134	CALL	BSIEXP
30A8	F5	0135	PUSH	AF
30A9	FE08	0136	CP	8
30AB	302F	0137	JR	NC,CH6
30AD	CD8734	0138	CALL	SELECT
30B0	C24A1E	0139	JP	NZ,BSFCE
30B3	F1	0140	POP	AF
30B4	7E	0141	LD	A,(HL)
30B5	CA1E1D	0142	JP	Z,BSCONT
30B8	111530	0143 CH5	LD	DE,BASIC
30BB	D5	0144	PUSH	DE
30BC	FEAD	0145	CP	BSCSV
30BE	2845	0146	JR	Z,SAVEB
30C0	FEA7	0147	CP	BSCLD
30C2	CA5231	0148	JP	Z,LOAD
30C5	FEBB	0149	CP	BSCNW
30C7	281D	0150	JR	Z,NEW
30C9	22E640	0151	LD	(BSHL),HL
30CC	2AB140	0152	LD	HL,(BSMEM)
30CF	23	0153	INC	HL
30D0	23	0154	INC	HL
30D1	E9	0155	JP	(HL)
30D2	D7	0156 GETN	RST	10H
30D3	0E00	0157	LD	C,0
30D5	2809	0158	JR	Z,GT1
30D7	CD1C2B	0159	CALL	BSIEXP
30DA	FE64	0160	CP	100
30DC	D24A1E	0161 CH6	JP	NC,BSFCE
30DF	4F	0162	LD	C,A
30E0	22E640	0163 GT1	LD	(BSHL),HL
30E3	79	0164	LD	A,C
30E4	B7	0165	OR	A
30E5	C9	0166	RET	
		0167 ;		
		0168 ;*** ADDED BASIC COMMANDS ***		
		0169 ;		
		0170 ;... '@NEW', '@NEW1', ETC. ...		
		0171 ;		

30E6	CDD230	0172	NEW	CALL	GETN
30E9	21C534	0173		LD	HL,NEWMSG
30EC	CD7935	0174		CALL	PRTSTG
30EF	61	0175		LD	H,C
30F0	CD5F32	0176		CALL	NEWA
30F3	F5	0177		PUSH	AF
30F4	E6B7	0178		AND	0B7H
30F6	2009	0179		JR	NZ,NW0
30F8	CDAF0F	0180		CALL	BSPRN
30FB	21CE34	0181		LD	HL,BYTEMSG
30FE	CD7935	0182		CALL	PRTSTG
3101	F1	0183	NW0	POP	AF
3102	C3FE31	0184		JP	LR6
		0185	:		
		0186	:... '@SAVE1', '@SAVE2', ETC. ...		
		0187	:		
3105	CDD230	0188	SAVEB	CALL	GETN
3108	281C	0189	SA0	JR	Z,SA1
310A	2B	0190		DEC	HL
310B	D7	0191		RST	10H
310C	281B	0192		JR	Z,SA2
310E	CF	0193		RST	8
310F	2C	0194		DB	','
3110	CD4731	0195		CALL	INTGR
3113	D5	0196		PUSH	DE
3114	CF	0197		RST	8
3115	2C	0198		DB	','
3116	CD4731	0199		CALL	INTGR
3119	D5	0200		PUSH	DE
311A	111530	0201		LD	DE,BASIC
311D	2819	0202		JR	Z,SA3
311F	CF	0203		RST	8
3120	2C	0204		DB	','
3121	CD4731	0205		CALL	INTGR
3124	2812	0206		JR	Z,SA3
3126	C39719	0207	SA1	JP	BSSYNE
3129	CDF81A	0208	SA2	CALL	BSFIND
312C	23	0209		INC	HL
312D	ED5BA440	0210		LD	DE,(BSBGNP)
3131	ED52	0211		SBC	HL,DE
3133	D5	0212		PUSH	DE
3134	E5	0213		PUSH	HL
3135	110000	0214		LD	DE,0
3138	21D634	0215	SA3	LD	HL,WRITMSG
313B	CD7935	0216		CALL	PRTSTG
313E	79	0217		LD	A,C
313F	C1	0218		POP	BC
3140	E1	0219		POP	HL
3141	CD8233	0220		CALL	SAVEA
3144	C3FB31	0221		JP	LR1
3147	C5	0222	INTGR	PUSH	BC
3148	CD022B	0223		CALL	BSINT
314B	2B	0224		DEC	HL
314C	D7	0225		RST	10H
314D	C1	0226		POP	BC
314E	22E640	0227		LD	(BSHL),HL
3151	C9	0228		RET	

```

                                0229 ;
                                0230 ;... 'eLOAD', 'eLOAD1', ETC. ...
                                0231 ;
3152 CDD230      0232 LOAD      CALL      GETN
3155 21DF34      0233 LOADA     LD         HL,READMSG
3158 CD7935      0234          CALL      PRTSTG
315B 2AA440      0235          LD         HL,(BSBGNP)
315E 116400      0236          LD         DE,100
3161 19          0237          ADD        HL,DE
3162 ED5BA040     0238          LD         DE,(BSMAXP)
3166 ED52        0239          SBC        HL,DE
3168 E5          0240          PUSH       HL
3169 2AA240      0241          LD         HL,(BSLIN)
316C 23          0242          INC        HL
316D 7C          0243          LD         A,H
316E B5          0244          OR         L
316F 280B        0245          JR         Z,LD1
3171 2AA440      0246          LD         HL,(BSBGNP)
3174 ED5BF940     0247          LD         DE,(BSENDP)
3178 37          0248          SCF
3179 ED52        0249          SBC        HL,DE
317B E3          0250          EX         (SP),HL
317C FDE1        0251 LD1      POP        IY
317E 210000      0252          LD         HL,0
3181 E5          0253          PUSH       HL
3182 61          0254          LD         H,C
3183 2EFF        0255          LD         L,OFFH
3185 79          0256          LD         A,C
3186 B7          0257          OR         A
3187 2001        0258          JR         NZ,LD2
3189 6F          0259          LD         L,A
318A CD3437      0260 LD2      CALL      MTON
318D CD4936      0261 LD3      CALL      RPREAM
3190 2068        0262          JR         NZ,LRTN
3192 7A          0263          LD         A,D
3193 B7          0264          OR         A
3194 28F7        0265          JR         Z,LD3
3196 FA8D31      0266          JP         M,LD3
3199 94          0267          SUB        H
319A A5          0268          AND        L
319B 20F0        0269          JR         NZ,LD3
319D CD0B37      0270          CALL      RDTWO
31A0 2058        0271          JR         NZ,LRTN
31A2 C0          0272          RET        NZ
31A3 C0          0273          RET        NZ
31A4 62          0274          LD         H,D
31A5 E5          0275          PUSH       HL
31A6 DDE1        0276          POP        IX
31A8 E5          0277          PUSH       HL
31A9 E1          0278          POP        HL
31AA 00          0279          NOP
31AB CD0B37      0280          CALL      RDTWO
31AE 204A        0281          JR         NZ,LRTN
31B0 62          0282          LD         H,D
31B1 7A          0283          LD         A,D
31B2 B5          0284          OR         L
31B3 283C        0285          JR         Z,LOADB

```

31B5	E5	0286	PUSH	HL
31B6	FDE1	0287	POP	IY
31B8	E5	0288	PUSH	HL
31B9	E1	0289	POP	HL
31BA	CD0B37	0290 LD4	CALL	RDTWO
31BD	203B	0291	JR	NZ,LRTN
31BF	5D	0292	LD	E,L
31C0	E1	0293	POP	HL
31C1	19	0294	ADD	HL,DE
31C2	3848	0295	JR	C,LR5
31C4	D5	0296	PUSH	DE
31C5	D5	0297	PUSH	DE
31C6	E1	0298	POP	HL
31C7	CD4536	0299	CALL	RBLOCK
31CA	202E	0300	JR	NZ,LRTN
31CC	D1	0301	POP	DE
31CD	FDE5	0302	PUSH	IY
31CF	E1	0303	POP	HL
31D0	7D	0304	LD	A,L
31D1	B4	0305	OR	H
31D2	2010	0306	JR	NZ,LD6
31D4	2AA240	0307	LD	HL,(BSLIN)
31D7	23	0308	INC	HL
31D8	7C	0309	LD	A,H
31D9	B5	0310	OR	L
31DA	211132	0311	LD	HL,RTNBS
31DD	280E	0312	JR	Z,LD7
31DF	217930	0313	LD	HL,AUTO
31E2	1809	0314	JR	LD7
31E4	3A8038	0315 LD6	LD	A,(BSSFT)
31E7	B7	0316	OR	A
31E8	2803	0317	JR	Z,LD7
31EA	211C32	0318	LD	HL,ABT
31ED	E3	0319 LD7	EX	(SP),HL
31EE	AF	0320	XOR	A
31EF	180A	0321	JR	LR1
31F1	00	0322 LOADB	NOP	
31F2	FDE3	0323	EX	(SP),IY
31F4	DD2AA440	0324	LD	IX,(BSBGNP)
31F8	18C0	0325	JR	LD4
31FA	E1	0326 LRTN	POP	HL
31FB	CD2537	0327 LR1	CALL	MTOFF
31FE	213735	0328 LR6	LD	HL,DONMSG
3201	CA7935	0329	JP	Z,PRTSTG
3204	CD3C35	0330	CALL	PRTERR
3207	1E2A	0331 BFD	LD	E,2AH
3209	03A219	0332	JP	BSERR
320C	3E20	0333 LR5	LD	A,20H
320E	B7	0334	OR	A
320F	18EA	0335	JR	LR1
3211	DD22F940	0336 RTNBS	LD	(BSENDP),IX
3215	2AA440	0337	LD	HL,(BSBGNP)
3218	E5	0338	PUSH	HL
3219	03E81A	0339	JP	BSFIX
321C	213135	0340 ABT	LD	HL,BRKMSG
321F	CD7935	0341	CALL	PRTSTG
3222	DDE5	0342	PUSH	IX

```

3224 E1          0343      POP      HL
3225 AF          0344      XOR      A
3226 ED52        0345      SBC      HL,DE
3228 D5          0346      PUSH     DE
3229 CDAF0F      0347      CALL     BSPRN
322C 3E2C        0348      LD       A,' ',' '
322E CD2A03      0349      CALL     BSPRTC
3231 E1          0350      POP      HL
3232 CDAF0F      0351      CALL     BSPRN
3235 3E2C        0352      LD       A,' ',' '
3237 CD2A03      0353      CALL     BSPRTC
323A FDE5        0354      PUSH     IY
323C E1          0355      POP      HL
323D CDAF0F      0356      CALL     BSPRN
3240 18C5        0357      JR       BFD
                0358 ;
                0359 ;*** SUBROUTINES FOR ASMB CALL ***
                0360 ; VECTORED FROM 3000H ETC.
                0361 ;
                0362 ; ALL RETURN WITH Z FOR CORRECT
                0363 ; NZ FOR ERROR WITH REG.A =
                0364 ; BIT0 --- WRITE WITHOUT DECAL
                0365 ; BIT1 --- USER HIT BREAK
                0366 ; BIT2 --- WRITE PASS EOT
                0367 ; BIT3 --- READ PARITY ERROR
                0368 ; BIT4 --- READ CHECKSUM ERROR
                0369 ; BIT5 --- RECORD TOO LONG
                0370 ; BIT6 --- VERIFY ERROR
                0371 ; BIT7 --- EOF READ
                0372 ;
                0373 ;... WIND TAPE TO BOT ...
                0374 ;
3242 C5          0375 REWIND  PUSH     BC
3243 D5          0376      PUSH     DE
3244 CD3437      0377      CALL     MTON
3247 CD4B37      0378      CALL     DELAY
324A 3A4038      0379 RWDL   LD       A,(BSBRK)
324D E604        0380      AND      4
324F 0F          0381      RRCA
3250 2007        0382      JR       NZ,RWDF
3252 ED78        0383      IN       A,(C)
3254 E604        0384      AND      4
3256 28F2        0385      JR       Z,RWDL
3258 AF          0386      XOR      A
3259 CD2537      0387 RWDF   CALL     MTOFF
325C D1          0388      POP      DE
325D C1          0389      POP      BC
325E C9          0390      RET
                0391 ;
                0392 ;... ERASE TAPE ...
                0393 ; H = STARTING FILE NUMBER
                0394 ; HL RETURNS NUMBER OF BYTES
                0395 ;
325F C5          0396 NEWA   PUSH     BC
3260 D5          0397      PUSH     DE
3261 CD3337      0398      CALL     MTONW
3264 E601        0399      AND      1

```


3266	206C	0400	JR	NZ,NW11
3268	7C	0401	LD	A,H
3269	FE02	0402	CP	2
326B	3018	0403	JR	NC,NW2
326D	CD4232	0404	CALL	REWIND
3270	2062	0405	JR	NZ,NW11
3272	3E85	0406	LD	A,85H
3274	321A40	0407	LD	(BSRAM),A
3277	ED79	0408	OUT	(C),A
3279	CD4B37	0409	CALL	DELAY
327C	2EFF	0410	LD	L,OFFH
327E	CD8635	0411	CALL	WRPRE
3281	2051	0412	JR	NZ,NW11
3283	1809	0413	JR	NW3
3285	CD4936	0414 NW2	CALL	RPREAM
3288	204A	0415	JR	NZ,NW11
328A	7A	0416	LD	A,D
328B	84	0417	ADD	H
328C	20F7	0418	JR	NZ,NW2
328E	0610	0419 NW3	LD	B,10H
3290	10FE	0420 NW4	DJNZ	NW4
3292	2E80	0421	LD	L,080H
3294	CD9235	0422	CALL	WPREAM
3297	C2D432	R 0423	JP	NZ,NW11
329A	1E5A	0424	LD	E,90
329C	26FF	0425 NW6	LD	H,OFFH
329E	54	0426	LD	D,H
329F	CD1936	0427	CALL	WRBIT
32A2	CA9C32	R 0428	JP	Z,NW6
32A5	0601	0429 NW7	LD	B,1
32A7	CD1936	0430	CALL	WRBIT
32AA	1D	0431	DEC	E
32AB	C2A532	R 0432	JP	NZ,NW7
32AE	0601	0433	LD	B,1
32B0	ED41	0434	OUT	(C),B
32B2	CD4936	0435 NW8	CALL	RPREAM
32B5	201D	0436	JR	NZ,NW11
32B7	7A	0437	LD	A,D
32B8	BD	0438	CP	L
32B9	20F7	0439	JR	NZ,NW8
32BB	21FFFF	0440	LD	HL,-1
32BE	23	0441 NW9	INC	HL
32BF	ED78	0442	IN	A,(C)
32C1	E604	0443	AND	4
32C3	200D	0444	JR	NZ,NW10
32C5	CD0D36	0445	CALL	RDBYTE
32C8	C2D232	R 0446	JP	NZ,NW10
32CB	F5	0447	PUSH	AF
32CC	F1	0448	POP	AF
32CD	14	0449	INC	D
32CE	3E08	0450	LD	A,8
32D0	28EC	0451	JR	Z,NW9
32D2	E6FB	0452 NW10	AND	OFBH
32D4	CD2537	0453 NW11	CALL	MTOFF
32D7	D1	0454	POP	DE
32D8	C1	0455	POP	BC
32D9	C9	0456	RET	

```

0457 ;
0458 ;... READ DATA BLOCK ...
0459 ; HL --> BUFFER
0460 ; BC = BUFFER LENGTH AT ENTRY
0461 ; RECORD LENGTH AT RETURN
0462 ;
32DA DDE5      0463 READ      PUSH      IX
32DC D5        0464          PUSH      DE
32DD E5        0465          PUSH      HL
32DE DDE1      0466          POP       IX
32E0 E5        0467          PUSH      HL
32E1 FDE5      0468          PUSH      IY
32E3 C5        0469          PUSH      BC
32E4 79        0470          LD        A,C
32E5 2F        0471          CPL
32E6 4F        0472          LD        C,A
32E7 78        0473          LD        A,B
32E8 2F        0474          CPL
32E9 47        0475          LD        B,A
32EA C5        0476          PUSH      BC
32EB FDE1      0477          POP       IY
32ED CD3437    0478          CALL      MTON
32F0 CD4936    0479 RLP      CALL      RPREAM
32F3 2025      0480          JR        NZ,RRTN
32F5 7A        0481          LD        A,D
32F6 B7        0482          OR        A
32F7 3E04      0483          LD        A,04H
32F9 201F      0484          JR        NZ,RRTN
32FB DD7E00    0485          LD        A,(IX)
32FE CD0B37    0486          CALL      RDTWO
3301 2017      0487          JR        NZ,RRTN
3303 62        0488          LD        H,D
3304 EB        0489          EX        DE,HL
3305 FD19      0490          ADD       IY,DE
3307 EB        0491          EX        DE,HL
3308 381B      0492          JR        C,BUFE
330A D8        0493          RET       C
330B 00        0494          NOP
330C E5        0495          PUSH      HL
330D E1        0496          POP       HL
330E CDA536    0497          CALL      RBLOCK
3311 2007      0498          JR        NZ,RRTN
3313 D1        0499          POP       DE
3314 FD19      0500          ADD       IY,DE
3316 FD23      0501          INC       IY
3318 FDE5      0502          PUSH      IY
331A CD2537    0503 RRTN     CALL      MTOFF
331D C1        0504          POP       BC
331E FDE1      0505          POP       IY
3320 E1        0506          POP       HL
3321 D1        0507          POP       DE
3322 DDE1      0508          POP       IX
3324 C9        0509          RET
3325 3E20      0510 BUFE     LD        A,20H
3327 B7        0511          OR        A
3328 18F0      0512          JR        RRTN
0513 ;

```

```

0514 ;... WRITE DATA RECORD ...
0515 ; HL --> BUFFER
0516 ; BC = BUFFER LENGTH
0517 ; D = DELAY TIME
0518 ;
332A DDE5 0519 WRTXEX PUSH IX
332C E5 0520 PUSH HL
332D DDE1 0521 POP IX
332F E5 0522 PUSH HL
3330 D5 0523 PUSH DE
3331 C5 0524 PUSH BC
3332 CD3337 0525 CALL MTONW
3335 2017 0526 JR NZ,WRTN
3337 2E00 0527 LD L,0
3339 CD9235 0528 CALL WPREAM
333C 2010 0529 JR NZ,WRTN
333E E1 0530 POP HL
333F E5 0531 PUSH HL
3340 DD7E00 0532 LD A,(IX)
3343 CD3B36 0533 CALL WRTWO
3346 2006 0534 JR NZ,WRTN
3348 00 0535 NOP
3349 E1 0536 POP HL
334A E5 0537 PUSH HL
334B CDC735 0538 CALL WBLOCK
334E CD2537 0539 WRTN CALL MTOFF
3351 E1 0540 POP HL
3352 D1 0541 POP DE
3353 E5 0542 PUSH HL
3354 CC4D37 0543 CALL Z,ONDLY
3357 C1 0544 POP BC
3358 E1 0545 POP HL
3359 DDE1 0546 POP IX
335B C9 0547 RET
335C D5 0548 WRITE PUSH DE
335D 163D 0549 LD D,61
335F CD2A33 0550 CALL WRTXEX
3362 D1 0551 POP DE
3363 C9 0552 RET
0553 ;
0554 ;... WRITE BUFFER & EOF ...
0555 ; HL --> BUFFER
0556 ; BC = BUFFER LENGTH
0557 ; A = FILL NUMBER
0558 ;
3364 F5 0559 WBE0F PUSH AF
3365 78 0560 LD A,B
3366 B1 0561 OR C
3367 C45C33 0562 CALL NZ,WRITE
336A F1 0563 POP AF
336B 1801 0564 JR WEOF
0565 ;
0566 ;... WRITE EOF MARK ...
0567 ; A = FILE NUMBER
0568 ;
336D 3D 0569 WEOFX DEC A
336E C5 0570 WEOF PUSH BC

```

336F	D5	0571	PUSH	DE
3370	E5	0572	PUSH	HL
3371	2F	0573	CPL	
3372	F680	0574	OR	80H
3374	6F	0575	LD	L,A
3375	CD3337	0576	CALL	MTONW
3378	CD8635	0577	CALL	Z,WRPRE
337B	CD2537	0578	CALL	MTOFF
337E	E1	0579	POP	HL
337F	D1	0580	POP	DE
3380	C1	0581	POP	BC
3381	C9	0582	RET	
		0583 ;		
		0584 ;...	WRITE PROGRAM FILE ...	
		0585 ;	HL = LOAD ADDRESS	
		0586 ;	DE = START ADDRESS	
		0587 ;	BC = LENGTH	
		0588 ;	A = FILE # (1 THRU 9)	
		0589 ;		
3382	DDE5	0590	SAVEA	PUSH IX
3384	FDE5	0591		PUSH IY
3386	E5	0592		PUSH HL
3387	D5	0593		PUSH DE
3388	C5	0594		PUSH BC
3389	E5	0595		PUSH HL
338A	DDE1	0596		POP IX
338C	E5	0597		PUSH HL
338D	6F	0598		LD L,A
338E	E5	0599		PUSH HL
338F	D5	0600		PUSH DE
3390	FDE1	0601		POP IY
3392	C5	0602		PUSH BC
3393	CD3337	0603		CALL MTONW
3396	2024	0604		JR NZ,SVEJ
3398	CD4936	0605	SV1	CALL RPREAM
339B	201F	0606		JR NZ,SVEJ
339D	7A	0607		LD A,D
339E	85	0608		ADD L
339F	20F7	0609		JR NZ,SV1
33A1	0610	0610		LD B,10H
33A3	10FE	0611	SV2	DJNZ SV2
33A5	CD9235	0612		CALL WFREAM
33A8	2012	0613		JR NZ,SVEJ
33AA	ED5A	0614		ADC HL,DE
33AC	DDE5	0615		PUSH IX
33AE	E1	0616		POP HL
33AF	CD3B36	0617		CALL WRTWO
33B2	2008	0618		JR NZ,SVEJ
33B4	ED5A	0619		ADC HL,DE
33B6	FDE5	0620		PUSH IY
33B8	E1	0621		POP HL
33B9	CD3B36	0622		CALL WRTWO
33BC	206F	0623	SVEJ	JR NZ,SVE
33BE	ED5A	0624		ADC HL,DE
33C0	ED5A	0625		ADC HL,DE
33C2	E1	0626		POP HL
33C3	CD3B36	0627		CALL WRTWO

33C6	2066	0628	JR	NZ,SVE1
33C8	E5	0629	PUSH	HL
33C9	E1	0630	POP	HL
33CA	00	0631	NOP	
33CB	CD0735	0632	CALL	WBLOCK
33CE	205E	0633	JR	NZ,SVE1
33D0	E1	0634	POP	HL
33D1	DDE1	0635	POP	IX
33D3	7D	0636	LD	A,L
33D4	2F	0637	CPL	
33D5	6F	0638	LD	L,A
33D6	CD8635	0639	CALL	WRPRE
33D9	2055	0640	JR	NZ,SVR
33DB	7D	0641	LD	A,L
33DC	2F	0642	CPL	
33DD	6F	0643	LD	L,A
33DE	3E01	0644	LD	A,1
33E0	ED79	0645	OUT	(C),A
33E2	CD4936	0646 SV3	CALL	RPREAM
33E5	2049	0647	JR	NZ,SVR
33E7	7A	0648	LD	A,D
33E8	BD	0649	CP	L
33E9	20F7	0650	JR	NZ,SV3
33EB	C0	0651	RET	NZ
33EC	E5	0652	PUSH	HL
33ED	E1	0653	POP	HL
33EE	CD0B37	0654	CALL	RDTWO
33F1	203D	0655	JR	NZ,SVR
33F3	C0	0656	RET	NZ
33F4	00	0657	NOP	
33F5	7D	0658	LD	A,L
33F6	DDE5	0659	PUSH	IX
33F8	E1	0660	POP	HL
33F9	BD	0661	CP	L
33FA	2065	0662	JR	NZ,ERR
33FC	7A	0663	LD	A,D
33FD	BC	0664	CP	H
33FE	2061	0665	JR	NZ,ERR
3400	CD0B37	0666	CALL	RDTWO
3403	202B	0667	JR	NZ,SVR
3405	C0	0668	RET	NZ
3406	00	0669	NOP	
3407	7D	0670	LD	A,L
3408	FDE5	0671	PUSH	IY
340A	E1	0672	POP	HL
340B	BD	0673	CP	L
340C	2053	0674	JR	NZ,ERR
340E	7A	0675	LD	A,D
340F	BC	0676	CP	H
3410	204F	0677	JR	NZ,ERR
3412	CD0B37	0678	CALL	RDTWO
3415	2019	0679	JR	NZ,SVR
3417	C0	0680	RET	NZ
3418	C0	0681	RET	NZ
3419	E5	0682	PUSH	HL
341A	E1	0683	POP	HL
341B	7D	0684	LD	A,L

341C	E1	0685	POP	HL
341D	E5	0686	PUSH	HL
341E	BD	0687	CP	L
341F	2040	0688	JR	NZ,ERR
3421	7A	0689	LD	A,D
3422	BC	0690	CP	H
3423	203C	0691	JR	NZ,ERR
3425	CD0D36	0692 SV4	CALL	RDBYTE
3428	2006	0693	JR	NZ,SVR
342A	C0	0694	RET	NZ
342B	180E	0695	JR	SV5
342D	E1	0696 SVE	POP	HL
342E	E1	0697 SVE1	POP	HL
342F	E1	0698	POP	HL
3430	CD2537	0699 SVR	CALL	MTOFF
3433	C1	0700	POP	BC
3434	D1	0701	POP	DE
3435	E1	0702	POP	HL
3436	FDE1	0703	POP	IY
3438	DDE1	0704	POP	IX
343A	C9	0705	RET	
343B	DD7E00	0706 SV5	LD	A,(IX)
343E	BA	0707	CP	D
343F	2020	0708	JR	NZ,ERR
3441	DD23	0709	INC	IX
3443	2B	0710	DEC	HL
3444	7C	0711	LD	A,H
3445	B5	0712	OR	L
3446	C22534	R 0713	JP	NZ,SV4
3449	CD0D36	0714	CALL	RDBYTE
344C	20E2	0715	JR	NZ,SVR
344E	0606	0716	LD	B,6
3450	10FE	0717 SV7	DJNZ	SV7
3452	CD0D36	0718	CALL	RDBYTE
3455	20D9	0719	JR	NZ,SVR
3457	3A1A40	0720	LD	A,(BSRAM)
345A	B7	0721	OR	A
345B	2802	0722	JR	Z,SV6
345D	3E0A	0723	LD	A,10
345F	18CF	0724 SV6	JR	SVR
3461	3E40	0725 ERR	LD	A,40H
3463	18CB	0726	JR	SVR
		0727 ;		
		0728 ;... FIND BEGINNING OF DATA FILE ...		
		0729 ; A = FILE NUMBER		
		0730 ;		
3465	C5	0731 FBOFX	PUSH	BC
3466	D5	0732	PUSH	DE
3467	F5	0733	PUSH	AF
3468	CD3437	0734	CALL	MTON
346B	CD4936	0735 OP1	CALL	RPREAM
346E	2005	0736	JR	NZ,OP2
3470	F1	0737	POP	AF
3471	F5	0738	PUSH	AF
3472	82	0739	ADD	D
3473	20F6	0740	JR	NZ,OP1
3475	CD2537	0741 OP2	CALL	MTOFF

3478	D1	0742	POP	DE
3479	D1	0743	POP	DE
347A	CC4D37	0744	CALL	Z,ONLY
347D	C1	0745	POP	BC
347E	C9	0746	RET	
347F	D5	0747	FBOF	PUSH DE
3480	163D	0748	LD	D,61
3482	CD6534	0749	CALL	FBOFX
3485	D1	0750	POP	DE
3486	C9	0751	RET	
		0752	:	
		0753	:	... SELECT DRIVE ...
		0754	:	AF = DRIVE NUMBER
		0755	:	
3487	C5	0756	SELECT	PUSH BC
3488	F6F0	0757	OR	OF0H
348A	4F	0758	LD	C,A
348B	ED78	0759	IN	A,(C)
348D	E608	0760	AND	8
348F	79	0761	LD	A,C
3490	C1	0762	POP	BC
3491	C0	0763	RET	NZ
3492	E5	0764	PUSH	HL
3493	2AB140	0765	LD	HL,(BSMEM)
3496	23	0766	INC	HL
3497	77	0767	LD	(HL),A
3498	E1	0768	POP	HL
3499	C9	0769	RET	
		0770	:	
		0771	:	...PRINT ERROR MESSAGE ...
		0772	:	A = ERROR CODE
		0773	:	
349A	E5	0774	ERROR	PUSH HL
349B	F5	0775		PUSH AF
349C	CD3C35	0776		CALL PRERR
349F	F1	0777		POP AF
34A0	E1	0778		POP HL
34A1	C9	0779		RET
		0780	:	
		0781	:	*** ** STRINGS *** **
		0782	:	
34A2	45584154	0783	HEAD	DB 'EXAT'
34A6	524F4E20	0784		DB 'RON '
34AA	53545249	0785		DB 'STRI'
34AE	4E475920	0786		DB 'NGY '
34B2	464C4F50	0787		DB 'FLOP'
34B6	50592056	0788		DB 'PY V'
34BA	45525349	0789		DB 'ERSI'
34BE	4F4E2034	0790		DB 'ON 4'
34C2	2E318D	0791		DB '.,1',CR+80H
34C5	45524153	0792	NEWMSG	DB 'ERAS'
34C9	494E472E	0793		DB 'ING.'
34CD	AE	0794		DB '.,'+80H
34CE	20425954	0795	BYTEMSG	DB 'BYT'
34D2	45532EAE	0796		DB 'ES.,'+80H
34D6	57524954	0797	WRITMSG	DB 'WRIT'
34DA	494E472E	0798		DB 'ING.'

34DE	AE	0799	DB	'.'+80H
34DF	52454144	0800	READMSG	DB 'READ'
34E3	494E472E	0801	DB	'ING.'
34E7	AE	0802	DB	'.'+80H
34E8	56455249	0803	VRFMSG	DB 'VERI'
34EC	46D9	0804	DB	'F', 'Y'+80H
34EE	50415249	0805	PERRMSG	DB 'PARI'
34F2	54D9	0806	DB	'T', 'Y'+80H
34F4	43484543	0807	CERRMSG	DB 'CHEC'
34F8	4B5355CD	0808	DB	'KSU', 'M'+80H
34FC	4F555420	0809	OMMSG	DB 'OUT '
3500	4F46204D	0810	DB	'OF M'
3504	454D4F52	0811	DB	'EMOR'
3508	D9	0812	DB	'Y'+80H
3509	54415045	0813	TTSMSG	DB 'TAPE'
350D	20544F4F	0814	DB	' TOO'
3511	2053484F	0815	DB	' SHO'
3515	52D4	0816	DB	'R', 'T'+80H
3517	57524954	0817	WPMSG	DB 'WRIT'
351B	452D5052	0818	DB	'E-PR'
351F	4F544543	0819	DB	'OTEC'
3523	54454480	0820	DB	'TE', 'D'+80H
3527	454FC6	0821	EOFMSG	DB 'EO', 'F'+80H
352A	20455252	0822	ERRMSG	DB ' ERR'
352E	4F528D	0823	DB	'OR', CR+80H
3531	42524541	0824	BRKMSG	DB 'BREA'
3535	4B8D	0825	DB	'K', CR+80H
3537	444F4E45	0826	DONEMSG	DB 'DONE'
353B	8D	0827	DB	CR+80H
		0828	:	
		0829	:	*** SUB-SUBROUTINES ***
		0830	:	
		0831	:	PRINT ERROR MESSAGE
		0832	:	
353C	CB47	0833	PRTERR	BIT 0, A
353E	211735	0834		LD HL, WPMSG
3541	2030	0835		JR NZ, PE1
3543	CB4F	0836	.BIT	1, A
3545	213135	0837		LD HL, BRKMSG
3548	202F	0838		JR NZ, PRTSTG
354A	CB57	0839		BIT 2, A
354C	210935	0840		LD HL, TTSMSG
354F	2022	0841		JR NZ, PE1
3551	CB5F	0842		BIT 3, A
3553	21EE34	0843		LD HL, PERRMSG
3556	201B	0844		JR NZ, PE1
3558	CB67	0845		BIT 4, A
355A	21F434	0846		LD HL, CERRMSG
355D	2014	0847		JR NZ, PE1
355F	CB6F	0848		BIT 5, A
3561	21FC34	0849		LD HL, OMMSG
3564	200D	0850		JR NZ, PE1
3566	CB77	0851		BIT 6, A
3568	21E834	0852		LD HL, VRFMSG
356B	2006	0853		JR NZ, PE1
356D	212735	0854		LD HL, EOFMSG
3570	CB7F	0855		BIT 7, A

3572	C8	0856	RET	Z
3573	CD7935	0857	CALL	PRTSTG
3576	212A35	0858	LD	HL,ERRMSG
		0859	;	
		0860	;	PRINT STRING POINTED BY HL
		0861	;	
3579	7E	0862	PRTSTG	LD A,(HL)
357A	23	0863	INC	HL
357B	F5	0864	PUSH	AF
357C	E67F	0865	AND	7FH
357E	CD2A03	0866	CALL	BSPRTC
3581	F1	0867	POP	AF
3582	17	0868	RLA	
3583	30F4	0869	JR	NC,PRTSTG
3585	C9	0870	RET	
		0871	;	
		0872	;	WRITE PREAMBLE, SYNC,
		0873	;	BYTE IN L AND TWO MORE
3586	CD9235	0874	WRPRE	CALL WPREAM
3589	C0	0875	RET	NZ
358A	C0	0876	RET	NZ
358B	0604	0877	LD	B,4
358D	10FE	0878	WEWE	DJNZ WEWE
358F	C33B36	0879	JP	WRTWO
		0880	;	
		0881	;	WRITE PREAMBLE, SYNC,
		0882	;	AND BYTE IN L
		0883	;	*** 14 ***
3592	3E85	0884	WPREAM	LD A,85H
3594	321A40	0885		LD (BSRAM),A
3597	ED79	0886		OUT (C),A
3599	CD4B37	0887		CALL DELAY
359C	C0	0888		RET NZ
359D	CD1436	0889		CALL WRBITS
35A0	C0	0890		RET NZ
35A1	00	0891		NOP
35A2	1800	0892		JR WRBLK
35A4	CD1936	0893	WRBLK	CALL WRBIT
35A7	C0	0894		RET NZ
35A8	C0	0895		RET NZ
35A9	0604	0896		LD B,4
35AB	10FE	0897	WRKWT2	DJNZ WRKWT2
35AD	3A1A40	0898		LD A,(BSRAM)
35B0	E67F	0899		AND 7FH
35B2	ED79	0900		OUT (C),A
35B4	00	0901		NOP
35B5	00	0902		NOP
35B6	0605	0903		LD B,5
35B8	1616	0904		LD D,SYNC
35BA	CDEC35	0905		CALL WRBYTE
35BD	C0	0906		RET NZ
35BE	55	0907		LD D,L
35BF	58	0908		LD E,B
35C0	0604	0909		LD B,4
35C2	C3EA35	R 0910		JP WRBY
35C5	00	0911	WRLOP	NOP
35C6	00	0912		NOP

```

0913 ;
0914 ;WRITE BLOCK OF HL
0915 ;POINTED BY IX
0916 ;*** 78/14 ***
35C7 DD5600 0917 WBLOCK LD D,(IX)
35CA CDEE35 0918 CALL WRBYT
35CD C0 0919 RET NZ
35CE DD23 0920 INC IX
35D0 2B 0921 DEC HL
35D1 7C 0922 LD A,H
35D2 B5 0923 OR L
35D3 20F0 0924 JR NZ,WRLOP
35D5 C0 0925 RET NZ
35D6 0601 0926 LD B,1
35D8 97 0927 SUB A
35D9 93 0928 SUB E
35DA 57 0929 LD D,A
35DB CDEC35 0930 CALL WRBYTE
35DE C0 0931 RET NZ
35DF C0 0932 RET NZ
35E0 00 0933 NOP
35E1 0604 0934 LD B,4
35E3 CDEC35 0935 CALL WRBYTE
35E6 C0 0936 RET NZ
35E7 C0 0937 RET NZ
35E8 0605 0938 LD B,5
35EA 00 0939 WRBY NOP
35EB 00 0940 NOP
0941 ;
0942 ;WRITE BYTE IN D
0943 ;*** 13B+37/14 ***
0944 ;*** 42/14 ***
35EC 10FE 0945 WRBYTE DJNZ WRBYTE
35EE 3A1A40 0946 WRBYT LD A,(BSRAM)
35F1 ED79 0947 OUT (C),A
35F3 7A 0948 LD A,D
35F4 83 0949 ADD A,E
35F5 5F 0950 LD E,A
35F6 0608 0951 LD B,8
35F8 23 0952 INC HL
35F9 CD1436 0953 CALL WRBITS
35FC 2B 0954 DEC HL
35FD C0 0955 RET NZ
35FE C0 0956 RET NZ
35FF 0602 0957 LD B,2
3601 10FE 0958 WRTWT DJNZ WRTWT
3603 3A1A40 0959 LD A,(BSRAM)
3606 F680 0960 OR 80H
3608 321A40 0961 LD (BSRAM),A
360B E67F 0962 AND 7FH
360D ED79 0963 OUT (C),A
360F AF 0964 XOR A
3610 C9 0965 RET
0966 ;
0967 ;WRITE B BITS IN D
0968 ;*** 116,99,75/28 ***
3611 DD7E00 0969 WRBITL LD A,(IX)

```

3614	ED78	0970	WRBITS	IN	A, (C)
3616	E605	0971		AND	5
3618	C0	0972		RET	NZ
3619	C5	0973	WRBIT	PUSH	BC
361A	3A1A40	0974		LD	A, (BSRAM)
361D	CB0A	0975		RRC	D
361F	3016	0976		JR	NC, WRZRO
3621	EE80	0977		XOR	80H
3623	ED79	0978		OUT	(C), A
3625	EE80	0979	WRCLK	XOR	80H
3627	00	0980		NOP	
3628	00	0981		NOP	
3629	BF	0982		CP	A
362A	0606	0983		LD	B, 6
362C	10FE	0984	WRBWT	DJNZ	WRBWT
362E	321A40	0985		LD	(BSRAM), A
3631	ED79	0986		OUT	(C), A
3633	C1	0987		POP	BC
3634	10DB	0988		DJNZ	WRBITL
3636	C9	0989		RET	
3637	00	0990	WRZRO	NOP	
3638	C32536	R 0991		JP	WRCLK
		0992		:	
		0993		:	WRITE TWO BYTES IN HL
		0994		:	*** 63/14 ***
363B	55	0995	WRTWO	LD	D, L
363C	CDEE35	0996		CALL	WRBYT
363F	C0	0997		RET	NZ
3640	00	0998		NOP	
3641	00	0999		NOP	
3642	00	1000		NOP	
3643	54	1001		LD	D, H
3644	0604	1002		LD	B, 4
3646	C3EC35	R 1003		JP	WRBYTE
		1004		:	
		1005		:	SEARCH PREAMBLE AND
		1006		:	SYNC, READ BYTE IN D
		1007		:	*** 81 ***
3649	CD5E36	1008	RPREAM	CALL	PR1
364C	C0	1009		RET	NZ
364D	7A	1010		LD	A, D
364E	D616	1011		SUB	SYNC
3650	C24936	R 1012		JP	NZ, RPREAM
3653	3C	1013		INC	A
3654	CD1937	1014		CALL	WAIT
3657	CD0D36	1015		CALL	RDBYTE
365A	321A40	1016		LD	(BSRAM), A
365D	C9	1017		RET	
365E	0614	1018	PR1	LD	B, 20
3660	CD1C37	1019	PR2	CALL	CHKBK
3663	ED78	1020		IN	A, (C)
3665	F26036	1021		JP	P, PR2
3668	CD1C37	1022	PR3	CALL	CHKBK
366B	ED78	1023		IN	A, (C)
366D	FA6836	1024		JP	M, PR3
3670	ED78	1025	PR4	IN	A, (C)
3672	FA5E36	1026		JP	M, PR1

3675	ED78	1027	PR5	IN	A,(C)
3677	F27536	1028		JP	P,PR5
367A	3E06	1029		LD	A,6
367C	3E06	1030		LD	A,6
367E	CD1937	1031		CALL	WAIT
3681	ED78	1032		IN	A,(C)
3683	F25E36	1033		JP	P,PR1
3686	ED78	1034	PR6	IN	A,(C)
3688	FA8636	1035		JP	M,PR6
368B	C38E36	R 1036		JP	PR7
368E	3E05	1037	PR7	LD	A,5
3690	CD1937	1038		CALL	WAIT
3693	10DB	1039		DJNZ	PR4
3695	04	1040		INC	B
3696	ED78	1041		IN	A,(C)
3698	F27536	1042		JP	P,PR5
369B	ED58	1043	PR8	IN	E,(C)
369D	FA9B36	1044		JP	M,PR8
36A0	3E00	1045		LD	A,0
36A2	C3DF36	R 1046		JP	RB1
		1047		:	
		1048		:	READ BLOCK OF HL
		1049		:	POINTED BY IX
		1050		:	*** 46/91 ***
36A5	CDCD36	1051	RBLOCK	CALL	RDBYTE
36A8	C0	1052		RET	NZ
36A9	C0	1053		RET	NZ
36AA	DD7200	1054		LD	(IX),D
36AD	DD23	1055		INC	IX
36AF	2B	1056		DEC	HL
36B0	00	1057		NOP	
36B1	E5	1058		PUSH	HL
36B2	E1	1059		POP	HL
36B3	7C	1060		LD	A,H
36B4	B5	1061		OR	L
36B5	C2A536	R 1062		JP	NZ,RBLOCK
36B8	CDCD36	1063		CALL	RDBYTE
36BB	C0	1064		RET	NZ
36BC	3E02	1065		LD	A,2
36BE	CD1937	1066		CALL	WAIT
36C1	CDCD36	1067		CALL	RDBYTE
36C4	C0	1068		RET	NZ
36C5	3A1A40	1069		LD	A,(BSRAM)
36C8	B7	1070		OR	A
36C9	C8	1071		RET	Z
36CA	3E10	1072		LD	A,10H
36CC	C9	1073		RET	
		1074		:	
		1075		:	READ BYTE IN D
		1076		:	*** 29/58 ***
36CD	ED78	1077	RDBYTE	IN	A,(C)
36CF	FAD636	1078		JP	M,RB0
36D2	3E08	1079		LD	A,08H
36D4	B7	1080		OR	A
36D5	C9	1081		RET	
36D6	ED58	1082	RB0	IN	E,(C)
36D8	FAD636	1083		JP	M,RB0

36DB	3A1A40	1084	LD	A, (BSRAM)
36DE	82	1085	ADD	A, D
36DF	321A40	1086	LD	(BSRAM), A
36E2	0608	1087	LD	B, 8
36E4	0608	1088	LD	B, 8
36E6	00	1089	NOP	
36E7	3E03	1090	LD	A, 3
36E9	CD1937	1091	CALL	WAIT
36EC	7B	1092	LD	A, E
36ED	2F	1093	CPL	
36EE	ED58	1094	IN	E, (C)
36F0	FAFB36	1095	JP	M, RB5
36F3	ED58	1096	IN	E, (C)
36F5	F2F336	1097	JP	P, RB4
36F8	C30337	R 1098	JP	RB6
36FB	ED58	1099	IN	E, (C)
36FD	FAFB36	1100	JP	M, RB5
3700	C30337	R 1101	JP	RB6
3703	AB	1102	XOR	E
3704	07	1103	RLCA	
3705	CB1A	1104	RR	D
3707	AF	1105	XOR	A
3708	10DC	1106	DJNZ	RB2
370A	C9	1107	RET	
		1108	;	
		1109	;	READ TWO BYTES INTO
		1110	;	L AND D
		1111	;	*** 46/58 ***
370B	CD0D36	1112	RDTWO	CALL RDBYTE
370E	C0	1113	RET	NZ
370F	6A	1114	LD	L, D
3710	00	1115	NOP	
3711	3E02	1116	LD	A, 2
3713	CD1937	1117	CALL	WAIT
3716	C30D36	R 1118	JP	RDBYTE
		1119	;	
		1120	;	*** 16A+43 ***
		1121	;	*** 48 ***
3719	3D	1122	WAIT	DEC A
371A	20FD	1123	JR	NZ, WAIT
371C	3A4038	1124	CHKBRK	LD A, (BSBRK)
371F	E604	1125	AND	4
3721	C8	1126	RET	Z
3722	0F	1127	RRCA	
3723	D1	1128	POP	DE
3724	C9	1129	RET	
3725	F5	1130	MTOFF	PUSH AF
3726	AF	1131	XOR	A
3727	ED79	1132	OUT	(C), A
3729	3A1240	1133	LD	A, (BSINTH)
372C	FEFB	1134	CP	CEI
372E	2801	1135	JR	Z, MT1
3730	FB	1136	EI	
3731	F1	1137	MT1	POP AF
3732	C9	1138	RET	
3733	F6	1139	MTONW	DB 0F6H
3734	AF	1140	MTON	XOR A

3735	E5	1141	PUSH	HL
3736	2AB140	1142	LD	HL, (BSMEM)
3739	23	1143	INC	HL
373A	4E	1144	LD	C, (HL)
373B	E1	1145	POP	HL
373C	2805	1146	JR	Z, MT2
373E	ED78	1147	IN	A, (C)
3740	E601	1148	AND	1
3742	C0	1149	RET	NZ
3743	3C	1150	INC	A
3744	ED79	1151	OUT	(C), A
3746	F3	1152	DI	
3747	1607	1153	LD	D, 7
3749	1802	1154	JR	ONDLY
374B	161F	1155	DELAY	LD D, 31
374D	AF	1156	ONDLY	XOR A
374E	B7	1157	DLP	OR A
374F	2004	1158	JR	NZ, DL1
3751	ED78	1159	IN	A, (C)
3753	E604	1160	AND	04H
3755	10F7	1161	DL1	DJNZ DLP
3757	15	1162	DEC	D
3758	20F4	1163	JR	NZ, DLP
375A	B7	1164	OR	A
375B	C9	1165	RET	
		1166	;	
		1167	****	*** COVER THEIR ASS ***
		1168	;	
375C	213640	1169	DEBNC	LD HL, BSKS
375F	010138	1170		LD BC, BSKEY
3762	1600	1171		LD D, 0
3764	0A	1172	DBLP	LD A, (BC)
3765	5F	1173		LD E, A
3766	AE	1174		XOR (HL)
3767	73	1175		LD (HL), E
3768	A3	1176		AND E
3769	2007	1177		JR NZ, DBDN
376B	14	1178		INC D
376C	2C	1179		INC L
376D	CB01	1180		RLC C
376F	F8	1181		RET M
3770	18F2	1182		JR DBLP
3772	5F	1183	DBDN	LD E, A
3773	C5	1184		PUSH BC
3774	0607	1185		LD B, 7
3776	CD6000	1186		CALL BSDLY
3779	C1	1187		POP BC
377A	0A	1188		LD A, (BC)
377B	A3	1189		AND E
377C	C8	1190		RET Z
377D	C3FB03	1191		JP BSKR
3780	(0000)	1192		END

Errors 0
Range Count 16

ABT	0340	0318
AUTO	0104	0057 0313
BASIC	0048	0143 0201
BFD	0331	0357
BRKMSG	0824	0340 0837
BSI	0049	0107
BSBGNP	0022	0105 0210 0235 0246 0324 0337
BSBRK	0036	0379 1124
BSCLD	0028	0147
BSCLR	0018	0093
BSCNW	0029	0149
BSCONT	0010	0049 0142
BSCSV	0027	0145
BSDLY	0033	1186
SENDP	0023	0247 0336
BSERR	0009	0332
BSEXE	0021	0113 0116
BSFCE	0011	0139 0161
BSFIND	0012	0104 0208
BSFIX	0026	0339
BSHL	0013	0048 0151 0163 0227
BSIEXP	0014	0134 0159
BSINC	0020	0118
BSINT	0005	0223
BSINTH	0015	1133
BSKEY	0035	1170
BSKI	0030	0098
BSKR	0032	1191
BSKS	0031	1169
BSLIN	0006	0241 0307
BSMAXP	0024	0238
BSMEM	0008	0080 0089 0152 0765 1142
BSFRN	0007	0180 0347 0351 0356
BSPRTC	0017	0349 0353 0866
BSRAM	0004	0077 0094 0407 0720 0885 0898 0946 0959 0961 0974 0985 1016 1069 1084 1086
BSRDY	0019	0103
BSRST2	0016	0079
BSSFT	0034	0315
BSSYNE	0025	0081 0083 0207
BUFE	0510	0492
BYTES	0795	0181
CEI	0003	1134
CERRMS	0807	0846
CH1	0117	0114
CH3	0126	0123
CH4	0133	0129
CH5	0143	0131
CH6	0161	0137
CHECK	0111	0078
CHKBRK	1124	1019 1022
CR	0001	0791 0823 0825 0827
DBDN	1183	1177
DBLP	1172	1182
DEBNC	1169	0097
DELAY	1155	0378 0409 0887
DL1	1161	1158

DLP	1157	1161	1163						
DONEMS	0826	0328							
EOFMSG	0821	0854							
ERR	0725	0662	0665	0674	0677	0688	0691	0708	
ERRMSG	0822	0858							
ERROR	0774	0055							
FBOF	0747	0046							
FBOFX	0731	0050	0749						
GETN	0156	0056	0172	0188	0232				
GT1	0163	0158							
HEAD	0783	0090							
INTGR	0222	0195	0199	0205					
LD1	0251	0245							
LD2	0260	0258							
LD3	0261	0265	0266	0269					
LD4	0290	0325							
LD6	0315	0306							
LD7	0319	0312	0314	0317					
LOAD	0232	0148							
LOADA	0233	0102							
LOADB	0322	0285							
LR1	0327	0221	0321	0335					
LR5	0333	0295							
LR6	0328	0184							
LRTN	0326	0262	0271	0281	0291	0300			
MT1	1137	1135							
MT2	1150	1146							
MTOFF	1130	0327	0387	0453	0503	0539	0578	0699	0741
MTON	1140	0260	0377	0478	0734				
MTONW	1139	0398	0525	0576	0603				
NEW	0172	0150							
NEWA	0396	0053	0176						
NEWMSG	0792	0173							
NW0	0183	0179							
NW10	0452	0444	0446						
NW11	0453	0400	0405	0412	0415	0423	0436		
NW2	0414	0403	0418						
NW3	0419	0413							
NW4	0420	0420							
NW6	0425	0428							
NW7	0429	0432							
NW8	0435	0439							
NW9	0441	0451							
OMMSG	0809	0849							
ONDLY	1156	0543	0744	1154					
OP1	0735	0740							
OP2	0741	0736							
PE1	0857	0835	0841	0844	0847	0850	0853		
PERRMS	0805	0843							
PR1	1018	1008	1026	1033					
PR2	1019	1021							
PR3	1022	1024							
PR4	1025	1039							
PR5	1027	1028	1042						
PR6	1034	1035							
PR7	1037	1036							
PR8	1043	1044							

PRTERR	0833	0330	0776
PRTSTG	0842	0091	0174 0182 0216 0234 0329 0341 0838 0857 0869
RBO	1082	1078	1083
RB1	1086	1046	
RB2	1089	1106	
RB4	1096	1097	
RB5	1099	1095	1100
RB6	1102	1098	1101
RBLOCK	1051	0299	0497 1062
RDBYTE	1077	0445	0692 0714 0718 1015 1051 1063 1067 1112 1118
RDTWO	1112	0270	0280 0290 0486 0654 0666 0678
READ	0463	0042	
READMS	0800	0233	
REWIND	0375	0041	0404
RLP	0479		
RFREAM	1008	0261	0414 0435 0479 0605 0646 0735 1012
RRTN	0503	0480	0484 0487 0498 0512
RTNBS	0336	0311	
RWDF	0387	0382	
RWDL	0379	0385	
SA0	0189	0133	
SA1	0207	0189	
SA2	0208	0192	
SA3	0215	0202	0206
SAVEA	0590	0045	0220
SAVEB	0188	0146	
SELECT	0756	0047	0138
ST1	0103	0096	
START	0071		
SV1	0605	0609	
SV2	0611	0611	
SV3	0646	0650	
SV4	0692	0713	
SV5	0706	0695	
SV6	0724	0722	
SV7	0717	0717	
SVE	0696	0623	
SVE1	0697	0628	0633
SVEJ	0623	0604	0606 0613 0618
SVR	0699	0640	0647 0655 0667 0679 0693 0715 0719 0724 0726
SYNC	0002	0904	1011
TTSMSG	0813	0840	
VRFMSG	0803	0852	
WAIT	1122	1014	1031 1038 1066 1091 1117 1123
WBEOF	0559	0054	
WBLOCK	0917	0538	0632
WEOF	0570	0052	0564
WEAFX	0569	0044	
WEWE	0878	0878	
WPMSG	0817	0834	
WPREAM	0884	0422	0528 0612 0874
WRBIT	0973	0427	0430 0893
WRBITL	0969	0988	
WRBITS	0970	0889	0953
WRBLK	0893	0892	
WRBW	0984	0984	
WRBY	0939	0910	

WRBYT	0946	0918	0996	
WRBYTE	0945	0905	0930	0935 0945 1003
WRCLK	0979	0991		
WRITE	0548	0043	0562	
WRITEX	0519	0051	0550	
WRITMS	0797	0215		
WRKWT2	0897	0897		
WRLOP	0911	0924		
WRPRE	0874	0411	0577	0639
WRTN	0539	0526	0529	0534
WRTWO	0995	0533	0617	0622 0627 0879
WRTWT	0958	0958		
WRZRO	0990	0976		

```

0001 ;*****
0002 ;
0003 ; STORAGE ALLOCATIONS FOR BUFFERS ETC.
0004 ;
0000' (7FFD') 0005 ORG      $+7500H
              0006 PREV    EQU      $+0AFDH ; VECTOR TO PREVIOUS MODUAL
0007 ;
7500' (0800) 0008 DS      8*256    ; UP TO 8 BUFFERS, 256 EACH
7D00' (0000) 0009 BUFFER DS      0
0010 ;
7D00' (0010) 0011 DRVCTR DS      16    ; DRIVE CONTROL BLOCK
              (0000) 0012 DS      EQU    0    ; DRIVE STATUS
              (0008) 0013 BN      EQU    8    ; BUFFER # FOR THAT DRIVE
0014 ;
7D10' (0020) 0015 BUFCTR DS      32    ; BUFFER CONTROL BLOCK
              (0000) 0016 BS      EQU    0    ; BUFFER STATUS
              (0008) 0017 RC      EQU    8    ; RECORD COUNT
              (0010) 0018 RL      EQU    16   ; RECORD LENGTH
              (0018) 0019 BP      EQU    24   ; BYTE POINTER
0020 ;
7D30' (0001) 0021 STAT    DS      1    ; ERROR CODE FOR FD ERRORS
0022 ;
0023 ;*****
0024 ;
0025 ; SYNTAX SCAN
0026 ;
7D31' 2AB140 0027 TAPE    LD      HL, (BSMEM)
7D34' 23      0028 INC     HL
7D35' 7E      0029 LD      A, (HL) ; DRIVE PORT #
7D36' E607    0030 AND     7
7D38' 4F      0031 LD      C, A
7D39' 0600    0032 LD      B, 0
7D3B' DD21007D' 0033 LD      IX, DRVCTR
7D3F' DD09    0034 ADD     IX, BC ; IX->DRVCTR
7D41' 2AE640 0035 LD      HL, (BSHL)
7D44' 7E      0036 LD      A, (HL)
7D45' FEB2    0037 CP      BSCPR
7D47' 2858    0038 JR      Z, PRINT
7D49' FE89    0039 CP      BSCIN
7D4B' CA137E' 0040 JP      Z, INPUT
7D4E' FEA2    0041 CP      BSCDP
7D50' 280D    0042 JR      Z, OPEN
7D52' FEA6    0043 CP      BSCLL
7D54' CA977E' 0044 JP      Z, CLOSE
7D57' FEB8    0045 CP      BSCCR
7D59' CA2A7F' 0046 JP      Z, CLEAR
7D5C' C3FD7F' 0047 JP      PREV ; PREVIOUS MODUAL
0048 ;
0049 ;*****
0050 ;
0051 ; @[#D]OPEN N
0052 ;
7D5F' DD7E00 0053 OPEN    LD      A, (IX+DS)
7D62' B7      0054 OR      A ; IS DRIVE FREE?
7D63' C2A1E   0055 JP      NZ, BSFCE ; NO, FC ERROR
7D66' FD21107D' 0056 LD      IY, BUFCTR ; YES
7D6A' 010008 0057 LD      BC, 0800H ; FIND A FREE BUFFER

```

```

7D6D' FD7E00      0058 L10      LD      A,(IY+BS)
7D70' B7          0059          OR      A
7D71' 2808        0060          JR      Z,L11
7D73' FD23'      0061          INC     IY
7D75' 0C          0062          INC     C
7D76' 10F5        0063          DJNZ   L10
7D78' C34A1E      0064 L12      JP      BSFCE ;NO FREE BUFFER, FC ERROR
7D7B' C5          0065 L11      PUSH    BC
7D7C' CD2D30      0066          CALL   ESFGTN ;GET FILE # N
7D7F' 28F7        0067          JR      Z,L12 ;N=0, FC ERROR
7D81' F5          0068          PUSH    AF
7D82' CD0F30      0069          CALL   ESFFBF ;FIND BEGINNING OF FILE N
7D85' C2967F'     0070          JP      NZ,BFD ;BAD, FD ERROR
7D88' DD360001    0071          LD      (IX+DS),1 ;CLAIM THIS DRIVE
7D8C' F1          0072          POP     AF
7D8D' FD7700      0073          LD      (IY+BS),A ;CLAIM THE BUFFER
7D90' FD360800    0074          LD      (IY+RC),0
7D94' FD361000    0075          LD      (IY+RL),0
7D98' FD361800    0076          LD      (IY+BP),0
7D9C' C1          0077          POP     BC
7D9D' DD7108      0078          LD      (IX+BN),C ;DRIVE TO BUFFER
7DA0' C9          0079          RET
0080 ;
0081 ;*****
0082 ;
0083 ; @[#DI]PRINT LIST-OF-EXPRESSIONS
0084 ;
7DA1' DD7E00      0085 PRINT    LD      A,(IX+DS)
7DA4' FE03        0086          CP      3
7DA6' 2809        0087          JR      Z,L20 ;WAS IT OUTPUT?
7DA8' FE01        0088          CP      1 ;JUST OPENED?
7DAA' C24A1E      0089          JP      NZ,BSFCE ;NO, FC ERROR
7DAD' DD360003    0090          LD      (IX+DS),3 ;MAKE IT OUTPUT
7DB1' 0600        0091 L20      LD      B,0
7DB3' DD4E08      0092          LD      C,(IX+BN)
7DB6' FD21107D'   0093          LD      IY,BUFCTR
7DBA' FD09        0094          ADD     IY,BC
7DBC' 41          0095          LD      B,C
7DBD' 04          0096          INC     B
7DBE' DD21007D'   0097          LD      IX,BUFFER
7DC2' 1100FF      0098          LD      DE,-256
7DC5' DD19        0099 L21      ADD     IX,DE
7DC7' 10FC        0100          DJNZ   L21
7DC9' FD4E18      0101          LD      C,(IY+BP)
7DCC' DD09        0102          ADD     IX,BC
7DCE' D7          0103 L22      RST     10H
7DCF' CD3723      0104          CALL   BSEXPR ;EVALUATE EXPRESSION
7DD2' 3AAF40      0105          LD      A,(BSVART)
7DD5' CDD27E'     0106          CALL   PUT ;PUT VALUE TYPE IN BUFFER
7DD8' 112141      0107          LD      DE,BSACC
7DDB' FE03        0108          CP      3
7DDD' 3809        0109          JR      C,L23 ;TYPE 2, INTEGER
7DDF' 2818        0110          JR      Z,L26 ;TYPE 3, STRING
7DE1' FE04        0111          CP      4
7DE3' 2803        0112          JR      Z,L23 ;TYPE 4, REAL
7DE5' 111D41      0113          LD      DE,BSACCD ;DOUBLE
7DE8' 47          0114 L23      LD      B,A

```

```

7DE9' 1A          0115 L24      LD      A,(DE)
7DEA' 13          0116          INC     DE
7DEB' CDD27E'     0117          CALL   PUT      ;PUT TYPE IN BUFFER
7DEE' 10F9        0118          DJNZ   L24
7DF0' 7E          0119 L25      LD      A,(HL)
7DF1' FE2C        0120          CP      ','      ;LIST SEPERATOR
7DF3' 28D9        0121          JR      Z,L22      ;MORE IN LIST
7DF5' 22E640      0122          LD      (BSHL),HL
7DF8' C9          0123          RET      ;END OF COMMAND
7DF9' E5          0124 L26      PUSH    HL      ;FIND STRING
7DFA' 1A          0125          LD      A,(DE)
7DFB' 6F          0126          LD      L,A
7DFC' 13          0127          INC     DE
7DFD' 1A          0128          LD      A,(DE)
7DFE' 67          0129          LD      H,A
7DFF' 7E          0130          LD      A,(HL)
7E00' CDD27E'     0131          CALL   PUT      ;STRING LENGTH IN BUFFER
7E03' 23          0132          INC     HL
7E04' 5E          0133          LD      E,(HL)
7E05' 23          0134          INC     HL
7E06' 56          0135          LD      D,(HL)
7E07' 21B540      0136          LD      HL,BSTMP+2
7E0A' 22B340      0137          LD      (BSTMP),HL
7E0D' E1          0138          POP     HL
7E0E' B7          0139          OR      A
7E0F' 28DF        0140          JR      Z,L25      ;NULL STRING
7E11' 18D5        0141          JR      L23      ;PUT STRING IN BUFFER
0142 ;
0143 ;*****
0144 ;
0145 ; @[#DIINPUT LIST-OF-VAREABLES
0146 ;
7E13' D07E00      0147 INPUT    LD      A,(IX+DS)
7E16' FE02        0148          CP      2
7E18' 2809        0149          JR      Z,L30      ;THIS WAS AN INPUT FILE
7E1A' FE01        0150          CP      1      ;OR WAS IT JUST OPENED?
7E1C' C24A1E      0151          JP      NZ,BSFCE      ;NO, FC ERROR
7E1F' DD360002    0152          LD      (IX+DS),2      ;MAKE IT AN INPUT
7E23' 0600        0153 L30      LD      B,0
7E25' DD4E08      0154          LD      C,(IX+BN)
7E28' FD21107D'   0155          LD      IY,BUFCTR
7E2C' FD09        0156          ADD     IY,BC
7E2E' 41          0157          LD      B,C
7E2F' 04          0158          INC     B
7E30' DD21007D'   0159          LD      IX,BUFFER
7E34' 1100FF      0160          LD      DE,-256
7E37' DD19        0161 L31      ADD     IX,DE
7E39' 10FC        0162          DJNZ   L31
7E3B' FD4E18      0163          LD      C,(IY+BP)
7E3E' DD09        0164          ADD     IX,BC
7E40' D7          0165 L32      RST      10H
7E41' 01717E'     0166          LD      BC,L35      ;RETURN ADDRESS
7E44' C5          0167          PUSH    BC
7E45' CD0D26      0168          CALL   BSVAR      ;FIND VARIABLE IN LIST
7E48' EB          0169          EX      DE,HL
7E49' 22DF40      0170          LD      (BSADD),HL
7E4C' EB          0171          EX      DE,HL

```

7E4D' D5	0172	PUSH	DE	
7E4E' E7	0173	RST	20H	;FIND VARIABLE TYPE
7E4F' F5	0174	PUSH	AF	
7E50' CDB07F'	0175	CALL	GETYP	;GET TYPE FROM BUFFER
7E53' 32AF40'	0176'	LD	(BSVART),A	
7E56' 112141	0177	LD	DE,BSACC	
7E59' FE03	0178	CP	3	
7E5B' 3809	0179	JR	C,L33	;TYPE 2, INTEGER
7E5D' 281B	0180	JR	Z,L36	;TYPE 3, STRING
7E5F' FE04	0181	CP	4	
7E61' 2803	0182	JR	Z,L33	;TYPE 4, REAL
7E63' 111D41	0183	LD	DE,BSACCD	;DOUBLE
7E66' 47	0184 L33	LD	B,A	
7E67' CDF67E'	0185 L34	CALL	GET	;GET LENGTH FROM BUFFER
7E6A' 12	0186	LD	(DE),A	
7E6B' 13	0187	INC	DE	
7E6C' 10F9	0188	DJNZ	L34	
7E6E' C3311F	0189 L37	JP	BSLET	;ASSIGN VALUE TO VARIABLE
7E71' 7E	0190 L35	LD	A,(HL)	;RETURN TO HERE
7E72' FE2C	0191	CP	','	;LIST SEPERATOR
7E74' 28CA	0192	JR	Z,L32	;MORE IN LIST
7E76' 22E640	0193	LD	(BSHL),HL	
7E79' C9	0194	RET		;END OF COMMAND
7E7A' E5	0195 L36	PUSH	HL	
7E7B' CDF67E'	0196	CALL	GET	;GET LENGTH FROM BUFFER
7E7E' 2AB340	0197	LD	HL,(BSTMP)	
7E81' 222141	0198	LD	(BSACC),HL	
7E84' 77	0199	LD	(HL),A	
7E85' ED5BA740	0200	LD	DE,(BSBUF)	
7E89' 23	0201	INC	HL	
7E8A' 73	0202	LD	(HL),E	
7E8B' 23	0203	INC	HL	
7E8C' 72	0204	LD	(HL),D	
7E8D' 23	0205	INC	HL	
7E8E' 22B340	0206	LD	(BSTMP),HL	
7E91' E1	0207	POP	HL	
7E92' B7	0208	OR	A	
7E93' 28D9	0209	JR	Z,L37	;NULL STRING
7E95' 18CF	0210	JR	L33	;GET STRING FROM BUFFER
	0211 ;			
	0212 ;*****			
	0213 ;			
	0214 ; @[#D]CLOSE			
	0215 ;			
7E97' D7	0216 CLOSE	RST	10H	
7E98' 22E640	0217	LD	(BSHL),HL	
7E9B' 0600	0218	LD	B,0	
7E9D' DD4E08	0219	LD	C,(IX+BN)	
7EA0' FD21107D'	0220	LD	IY,BUFCTR	
7EA4' FD09	0221	ADD	IY,BC	
7EA6' DD7E00	0222	LD	A,(IX+DS)	
7EA9' B7	0223	OR	A	
7EAA' C8	0224	RET	Z	;DRIVE NOT ACTIVE
7EAB' FE03	0225	CP	3	
7EAD' 201A	0226	JR	NZ,L41	;DRIVE WAS FOR INPUT
7EAF' FD7E18	0227	LD	A,(IY+BP)	;WAS FOR OUTPUT
7EB2' E5	0228	PUSH	HL	

```

7EB3' 41          0229      LD      B,C
7EB4' 04          0230      INC     B
7EB5' 21007D'     0231      LD      HL,BUFFER
7EB8' 1100FF     0232      LD      DE,-256
7EBB' 19          0233 L40     ADD     HL,DE
7EBC' 10FD       0234      DJNZ    L40
7EBE' 4F          0235      LD      C,A
7EBF' FD7E00     0236      LD      A,(IY+BS)      ;A=FILE #
7EC2' CD2730     0237      CALL    ESFWBF ;WRITE BUFFER AND EOF
7EC5' C2967F'    0238      JP      NZ,BFD ;DID NOT WORK, FD ERROR
7EC8' E1          0239      POP     HL
7EC9' DD360000   0240 L41     LD      (IX+DS),0      ;SET DRIVE FREE
7ECD' FD360000   0241      LD      (IY+BS),0      ;SET BUFFER FREE
7ED1' C9          0242      RET
0243 ;
0244 ;*****
0245 ;
0246 ; PUT A BYTE IN BUFFER
0247 ;
7ED2' DD7700     0248 PUT     LD      (IX),A
7ED5' DD23       0249      INC     IX
7ED7' FD3418     0250      INC     (IY+BP)
7EDA' C0         0251      RET     NZ      ;NOT FULL, JOB DONE
7EDB' E5         0252      PUSH    HL      ;BUFFER FULL
7EDC' C5         0253      PUSH    BC
7EDD' F5         0254      PUSH    AF
7EDE' 0100FF     0255      LD      BC,-256
7EE1' DD09       0256      ADD     IX,BC
7EE3' DDE5       0257      PUSH    IX
7EE5' E1         0258      POP     HL
7EE6' 010001     0259      LD      BC,256
7EE9' CD0630     0260      CALL    ESFWRT ;WRITE BUFFER ON TAPE
7EEC' C2967F'    0261      JP      NZ,BFD ;OH, SHIT
7EEF' FD3408     0262      INC     (IY+RC)
7EF2' F1         0263      POP     AF
7EF3' C1         0264      POP     BC
7EF4' E1         0265      POP     HL
7EF5' C9         0266      RET
0267 ;
0268 ;*****
0269 ;
0270 ; GET A BYTE FROM BUFFER
0271 ;
7EF6' C5         0272 GET     PUSH    BC
7EF7' FD7E18     0273      LD      A,(IY+BP)
7EFA' B7         0274      OR      A
7EFB' 2025       0275      JR      NZ,L62 ;POINTER>0
7EFD' E5         0276      PUSH    HL      ; =0, BUFFER EMPTY
7EFE' DDE5       0277      PUSH    IX
7F00' E1         0278      POP     HL
7F01' 010001     0279      LD      BC,256
7F04' CD0330     0280      CALL    ESFRD ;READ A RECORD FROM TAPE
7F07' C2917F'    0281      JP      NZ,ERR ;OUT OF DATA OR REAL BAD?
7F0A' FD7110     0282      LD      (IY+RL),C
7F0D' FD3408     0283      INC     (IY+RC)
7F10' E1         0284      POP     HL
7F11' DD7E00     0285 L60     LD      A,(IX) ;GET THE BYTE

```

```

7F14' DD23          0286      INC      IX
7F16' FD3418        0287      INC      (IX+BP)
7F19' 2005          0288      JR       NZ,L61
7F1B' 0100FF        0289      LD       BC,-256
7F1E' DD09          0290      ADD      IX,BC
7F20' C1            0291 L61     POP      BC
7F21' C9            0292      RET
7F22' FDBE10        0293 L62     CP       (IX+RL) ;MAY BE END OF DATA
7F25' CAA022        0294      JP       Z,BSODE ;YES, OD ERROR
7F28' 18E7          0295      JR       L60      ;NO, GO AHEAD
0296 ;
0297 ;*****
0298 ;
0299 ; @CLEAR [N]
0300 ;
7F2A' CD2D30        0301 CLEAR   CALL     ESFGTN ;GET VALUE OF N
7F2D' F5            0302      PUSH    AF      ;SAVE IT
7F2E' FE09          0303      CP       9
7F30' D24A1E        0304      JP       NC,BSFCE ;N>8, FC ERROR
7F33' 22E640        0305      LD       (BSHL),HL
7F36' 0608          0306      LD       B,B
7F38' DD21007D'     0307      LD       IX,DRVCTR
7F3C' DD360000      0308 L71     LD       (IX+DS),0 ;FREE ALL DRIVES
7F40' DD7E10        0309      LD       A,(IX+BUFCTR-DRVCTR)
7F43' 3C            0310      INC      A ;IS BUFFER ALLOCATED?
7F44' 2804          0311      JR       Z,L77 ;NO, DON'T TOUCH IT
0312 ;ELSE, FREE THIS BUFFER
7F46' DD361000      0313      LD       (IX+BUFCTR-DRVCTR),0
7F4A' DD23          0314 L77     INC      IX
7F4C' 10EE          0315      DJNZ    L71
7F4E' 2AB140        0316      LD       HL,(BSMEM)
7F51' 23            0317      INC      HL
7F52' 36F0          0318      LD       (HL),0F0H ;SET DRIVE # TO 0
7F54' F1            0319      POP      AF ;OPTIONAL N
7F55' C8            0320      RET      Z ;N=0 OR NOT THERE
7F56' 21177D'     0321      LD       HL,BUFCTR+BS+7
7F59' 0608          0322      LD       B,B
7F5B' 36FF          0323 L72     LD       (HL),0FFH
7F5D' B8            0324      CP       B
7F5E' 3802          0325      JR       C,L73
7F60' 3600          0326      LD       (HL),0 ;SET N OF THEM FREE
7F62' 2B            0327 L73     DEC      HL
7F63' 10F6          0328      DJNZ    L72
7F65' 21FB7C'     0329      LD       HL,BUFFER-5 ;COMPUTE HIGHEST
7F68' 1100FF        0330      LD       DE,-256 ; ADDR FOR BASIC
7F6B' 47            0331      LD       B,A
7F6C' 19            0332 L74     ADD      HL,DE
7F6D' 10FD          0333      DJNZ    L74
7F6F' 22B140        0334      LD       (BSMEM),HL ;GIVE IT TO BASIC
7F72' 31F07C'     0335      LD       SP,BUFFER-16
7F75' 23            0336      INC      HL
7F76' 36F0          0337      LD       (HL),0F0H ;PORT # AND
7F78' 23            0338      INC      HL
7F79' 36C3          0339      LD       (HL),0C3H ;JUMP INSTR.
7F7B' 23            0340      INC      HL
7F7C' 22837F'     0341      LD       (L75+1),HL
7F7F' 21317D'     0342      LD       HL,TAPE

```



```

7F82' 220000      0343 L75      LD      (0),HL
7F85' 113200      0344          LD      DE,50
7F88' 0D831E      0345          CALL   BSCLR          ;CLEAR 50
7F8B' 2AE640      0346          LD      HL,(BSHL)
7F8E' 031E1D      0347          JP      BSCONT          ;BACK TO BASIC
0348 ;*****
0349 ;
0350 ; OUT OF DATA OR BAD FILE DATA
0351 ;
7F91' FE04        0352 ERR      CP      04H          ;IS IT EOF ERROR?
7F93' CAA022      0353          JP      Z,BSODE        ;YES, OD ERRER
0354 ;
7F96' 5F          0355 BFD      LD      E,A          ;BAD FILE DATA
7F97' 32307D'     0356          LD      (STAT),A      ;SAVE ERROR CODE
7F9A' 2AF040      0357          LD      HL,(BSONEF)    ;ON ERROR GOTO ?
7F9D' 7C          0358          LD      A,H
7F9E' B5          0359          OR      L
7F9F' 2806        0360          JR      Z,L81
7FA1' 3AF240      0361          LD      A,(BSONEF)      ;FLAG SET?
7FA4' B7          0362          OR      A
7FA5' 2804        0363          JR      Z,FDE
7FA7' 7B          0364 L81      LD      A,E          ;PRINT ESF ERROR MSG
7FA8' 0D2A30      0365          CALL   ESFPRI
7FAB' 1E2A        0366 FDE      LD      E,2AH        ;FD ERROR CODE
7FAD' 03A119      0367          JP      BSERR
0368 ;
0369 ;*****
0370 ;
0371 ; GET DATA TYPE FROM BUFFER
0372 ;
7FB0' 0DF67E'     0373 GETYP    CALL   GET          ;GET A BYTE
7FB3' FE02        0374          CP      2
7FB5' 38F4        0375          JR      C,FDE          ;TYPE MUST >= 2
7FB7' FE05        0376          CP      5
7FB9' D8          0377          RET      C          ; AND < 5
7FBA' FE08        0378          CP      8
7FBC' 08          0379          RET      Z          ; OR = 8
7FBD' 18EC        0380          JR      FDE
0381 ;
0382 ;*****
0383 ;
0384 ; LINKS TO BASIC INTERPRETER
0385 ;
(40E6)      0386 BSHL      EQU      40E6H
(19A1)      0387 BSERR     EQU      19A1H
(2337)      0388 BSEXPR    EQU      2337H
(4121)      0389 BSACC     EQU      4121H
(411D)      0390 BSACCD    EQU      411DH
(40B1)      0391 BSMEM     EQU      40B1H
(1E4A)      0392 BSFCE     EQU      1E4AH
(40AF)      0393 BSVART    EQU      40AFH
(260D)      0394 BSVAR     EQU      260DH
(40DF)      0395 BSADD     EQU      40DFH
(1F31)      0396 BSLET     EQU      1F31H
(40B3)      0397 BSTMP     EQU      40B3H
(40A7)      0398 BSBUF     EQU      40A7H
(1E83)      0399 BSCLR     EQU      1E83H

```

```

(1D1E)      0400 BSCONT EQU      1D1EH
(22A0)      0401 BCODE  EQU      22A0H
(40F0)      0402 BSONEL EQU      40F0H
(40F2)      0403 BSONEF EQU      40F2H
0404 ;
0405 ;*****
0406 ;
0407 ; TOKENS FOR BASIC
0408 ;
(00B2)      0409 BSCPR  EQU      0B2H
(00B9)      0410 BSCIN  EQU      0B9H
(00A2)      0411 BSCOP  EQU      0A2H
(00A6)      0412 BSCCL  EQU      0A6H
(00B8)      0413 BSCCR  EQU      0B8H
0414 ;
0415 ;*****
0416 ;
0417 ; LINKS TO ESF FIRMWARE
0418 ;
(3003)      0419 ESFRED EQU      3003H
(3006)      0420 ESFWRT EQU      3006H
(300F)      0421 ESFFBF EQU      300FH
(3027)      0422 ESFWBF EQU      3027H
(302D)      0423 ESFGTN EQU      302DH
(302A)      0424 ESFPRI EQU      302AH
0425 ;
7FBF (0000) 0426      END

```

Errors 0

Program Length 7FBF (32703)

BFD	0355	0070	0238	0261				
BN	0013	0078	0092	0154	0219			
BP	0019	0076	0101	0163	0227	0250	0273	0287
BS	0016	0058	0073	0236	0241	0321		
BSACC	0389	0107	0177	0198				
BSACCD	0390	0113	0183					
BSADD	0395	0170						
BSBUF	0398	0200						
BSCCL	0412	0043						
BSCCR	0413	0045						
BSCIN	0410	0039						
BSCLR	0399	0345						
BSCONT	0400	0347						
BSCOP	0411	0041						
BSCPR	0409	0037						
BSERR	0387	0367						
BSEXPR	0388	0104						
BSFCE	0392	0055	0064	0089	0151	0304		
BSHL	0386	0035	0122	0193	0217	0305	0346	
BSLET	0396	0189						
BSMEM	0391	0027	0316	0334				
BSODE	0401	0294	0353					
BSONEF	0403	0361						
BSONEL	0402	0357						
BSTMP	0397	0136	0137	0197	0206			
BSVAR	0394	0163						
BSVART	0393	0105	0176					
BUFCTR	0015	0056	0093	0155	0220	0309	0313	0321
BUFFER	0009	0097	0159	0231	0329	0335		
CLEAR	0301	0046						
CLOSE	0216	0044						
DRVCTR	0011	0033	0307	0309	0313			
DS	0012	0053	0071	0085	0090	0147	0152	0222 0240 0308
ERR	0352	0281						
ESFFBF	0421	0069						
ESFGTN	0423	0066	0301					
ESFPRI	0424	0365						
ESFRED	0419	0280						
ESFWBF	0422	0237						
ESFWRT	0420	0260						
FDE	0366	0363	0375	0380				
GET	0272	0185	0196	0373				
GETYP	0373	0175						
INPUT	0147	0040						
L10	0058	0063						
L11	0065	0060						
L12	0064	0067						
L20	0091	0087						
L21	0099	0100						
L22	0103	0121						
L23	0114	0109	0112	0141				
L24	0115	0118						
L25	0119	0140						
L26	0124	0110						
L30	0153	0149						
L31	0161	0162						
L32	0165	0192						

ESF DATA I/O FOR TRS-80
CROSS REFERENCE LISTING

L33	0184	0179	0182	0210
L34	0185	0188		
L35	0190	0166		
L36	0195	0180		
L37	0189	0209		
L40	0233	0234		
L41	0240	0226		
L60	0285	0295		
L61	0291	0288		
L62	0293	0275		
L71	0308	0315		
L72	0323	0328		
L73	0327	0325		
L74	0332	0333		
L75	0343	0341		
L77	0314	0311		
L81	0364	0360		
OPEN	0053	0042		
PREV	0006	0047		
PRINT	0085	0038		
PUT	0248	0106	0117	0131
RC	0017	0074	0262	0283
RL	0018	0075	0282	0293
STAT	0021	0356		
TAPE	0027	0342		