FLEX
For the TRS-80 Color Computer

FRANK
HOGG
LABORATORY

THE REGENCY TOWER • 770 JAMES ST • SYRACUSE NY 13203 • (315) 474 7856

Technical Systems Consultants, Inc.
P.O. Box 2570

FLEX
For the TRS-80 Color Computer

FRANK
HOGG
LABORATORY

THE REGENCY TOWER • 770 JAMES ST • SYRACUSE NY 13203 • (315) 474-7856

Technical Systems Consultants, Inc.
P.O. Box 2570
West Lafayette, Indiana 47906

FHL
Color
FLEX

READ THIS FIRST

5.0:4 Color FLEX has several new features. First put the supplied disk in drive 0 and type RUN"FLEX . After about 30 seconds the screen will switch to Hi-Resolution and the FHL Color FLEX logo with a date prompt will be displayed. Answer the date prompt with today's date. In a few moments the screen will clear and a status line will appear on the bottom of the screen with the FHL logo and the date in it, then an example of the character set will be displayed on the screen. Finally the three plus sign prompt will appear.

Everything that happens from the time you entered the date until the three plus sign prompt is because of the "STARTUP" file being executed as per the documentation. If you list the startup file, i.e. LIST STARTUP, you will see PUTSTAT:LIST ALPHA. PUTSTAT is a trivial program that sets a status line at the bottom of the screen and then puts the logo and the date in it. LIST ALPHA is just that, LIST being a program that lists a file, and ALPHA being a text file that contains just exactly what you see displayed.

New features of version 5.0:4 include the DISPLAY command and an extensive revision of the EXT command bringing with it true hardware handshaking! Along with this, the INT command has been modified. Smooth scrolling and variable scroll rates along with a new SETUP command to change your screen from a white background to green (eliminating the need to change it by manually adjusting the VDG mode as in previous versions) are also included and explained in the "Features of FHL Color FLEX" section of this manual. Control-W is now used to generate Status lines instead of the Control-C used previously.

These are just some of the features of this new version. Read in the manual about how to create your own STARTUP file and the use of commands in it.

A NOTE ABOUT STATUS LINES

When you use status lines you are changing the configuration of the terminal. This means that if you were to run a program like our 'ED' which uses cursor addressing and depends on the configuration of the terminal for proper running, then you have to reconfigure those programs. Or, before running any of these programs, just type a CTRL D which will remove any status lines. The same can be said for protected lines on the screen. It's no big problem, as a matter of fact it can be rather entertaining to see what happens when you do this. At any rate, have fun with the new features of FLEX.

This version of FLEX for the Color Computer is the most powerful available, with more features than any other currently available. Just wait 'till you see what we're working on next.

FHL

(*) These commands are provided by Frank Hogg Laboratory, Inc., not TSC. Contact FHL if you have any problems with them.

NOTE: IF THERE IS A 'READ-ME.TXT' FILE ON THE SUPPLIED DISK THEN LIST IT OUT USING:

```
LIST 0.READ-ME
```

(*) These commands are provided by Frank Hogg Laboratory, Inc. - not TSC. Contact FHL if you have any problems with them.

MANUAL REVISION HISTORY

Revision	Date	Change
A	5/27/82	Original Release, V5.0
B	7/14/82	Correct typos, add INCHNE to Appendix E, Prog. Guide page 53 addition.
C	11/24/82	Rewrite and update to Version 5.0:2
D	3/18/83	Addition of ISM and TED.
E	6/13/83	Rewrite and update to Version 5.0:4
F	9/9/83	Include omissions to previous rewrite. Correct typos.

Introduction to FHL Color FLEX

1. Introduction

FHL Color FLEX, licensed from Technical Systems Consultants, Inc. by Frank Hogg Laboratory, Inc., is an enhanced version of their FLEX Disk Operating System, for the Radio Shack TRS-80 Color Computer. FHL Color FLEX allows the Color Computer to be used with a wide array of hardware and software products, far beyond the capabilities of the basic system as supplied by Radio Shack. For a list of some of the software products available for FLEX, see Appendix D.

The rest of Section 1 details the minimum hardware configuration required to run FHL Color FLEX. Section 2 of this document describes some of the features of FHL Color FLEX. Features of FLEX as supplied by TSC are not described here; some articles are referenced in Appendix A. Section 3 provides a tutorial on the use of some of the commands provided with FHL Color FLEX.

Minimum Hardware Configuration.

To run FHL Color FLEX you must have a TRS80 Color Computer with Disk Extended Color Basic and at least one disk drive. This implies that the Extended Color Basic ROM is installed in the Color Computer. In addition, 64K dynamic RAMs must be installed in the Color Computer, which must be configured as a 32K Color Computer. Finally, a small hardware modification (Revision F boards do not require modification), is required inside the case of the Color Computer to allow the ROMs to be disabled under software control, allowing the full 64K of RAM (minus the top 256 bytes) to be accessed.

If you do not have a Color Computer, you may purchase one already modified and ready to run FHL Color FLEX. Some vendors are listed in Appendix B. If you have a Color Computer that does not have 32K, Radio shack will upgrade it for you. The RAM chips used by Radio Shack will probably be full 64K RAMs, but they must be tested after performing the modification, and one or two may need to be replaced. Alternatively, if your computer is a recent revision (D,E or F), a 64K upgrade kit is available.

If you have a 32K Color Computer, Appendix C tells you how to perform the modification to use the full 64K.

Running FHL Color FLEX on the modified 64K Color Computer with the Radio Shack disk system and Disk Extended Color Basic is accomplished by inserting the system disk in drive zero and entering:

```
    RUN "FLEX"
```

The disk motors should start and after about 10 seconds, the following should appear on the terminal.

```
FHL COLOR FLEX Vx.x:x
DATE (MM,DD,YY)?

+++
```

The name FLEX identifies the operating system and the x.x:x will be the version number of the operating system. At this time the current date should be entered, such as 5,26,82 or 5 26 82 or 5/26/82. The FLEX prompt is the three plus signs (+++), and will always be present when the system is ready to accept an operator command. The '+++' should become a familiar sight and signify that FLEX is ready to work for you!

2. Features of FHL Color FLEX

Disk Drive Support

The disk drive supplied with the Color Computer disk system by Radio Shack is a 35-track, single-sided, double-density drive with a stepping rate of 30 milliseconds. It is possible to purchase an 80-track, double-sided, double-density drive with a stepping rate of 6ms. This disk will store more than four times as much information as the Radio Shack disk, and will perform track seeking operations five times faster. FHL Color FLEX allows these high-performance drives to be used with the Color Computer.

A section of memory in the FLEX operating system is reserved for a Drive Configuration Table. This table contains information about each drive in the system, up to a total of four drives. (Only three drives may be used if double-sided drives are included.) The information about each drive in the table includes whether or not the drive exists, whether or not it is double-sided or double-density, how many tracks it has, and what its stepping rate is. This means that not only may you use high-performance drives in the system, but also you may mix high-performance drives with standard Radio Shack drives in any combination. Another feature of the Drive Configuration Table allows the drive number used by FLEX (the logical drive number) to be different from the drive number according to the wiring (the physical drive number). Care must be exercised using this feature, because the number switching is not in effect during boot-up, and because different logical drives may be assigned to the same physical drive and vice-versa, possibly confusing the FLEX file management system.

FHL Color FLEX contains another feature rarely found in FLEX systems for five-inch disks. A five-inch drive has no provision for telling the computer that there is no disk in the drive, or that the door is open. This means that on most five-inch FLEX systems the software will hang up if you accidentally access a drive which does not exist, or which exists but does not contain a disk. In FHL Color FLEX, an access to a nonexistent drive will immediately return an error (because the Drive Configuration Table indicates that the drive doesn't exist.) A time limit is imposed on accesses to existing drives so that if the drive is not ready, an error will occur.

Another feature of the disk package in FHL Color FLEX is that the time delay before motor shutoff occurs is adjustable.

Keyboard Features.

Every ASCII code may be generated from the Color Computer keyboard under FHL Color FLEX. To allow for this, some of the keys must serve dual functions. Keycodes commonly used in FLEX have been assigned so that they may be easily generated.

If the up-arrow is struck alone, the cursor is moved up one line. However, if the shift key is held down, the up-arrow key serves as the CTRL key. In the unshifted mode, the break key generates an escape code. In control mode however, the break key is another case-shifting key, called super-shift. The super-shift feature and the CTRL feature allow all codes to be generated. Appendix E contains a list of the codes generated by each of the keys, in each mode.

Some special function keys are worthy of note. A shift-zero toggles the alpha-lock function: If alpha-lock is on, only upper-case letters are generated.

The number of times that the keyboard is scanned for debouncing purposes is adjustable, using the SETUP command. This feature allows the debouncing operation to be optimized for faster typists or, on the other hand, for bouncier keyboards.

Display Features

The display package included in FHL Color FLEX contains many features and adjustable parameters to enhance the capabilities of the Color Computer display. The user may select either a blinking block or steady underline cursor. This is done by typing SETUP TCC. The cursor may also be disabled. Screen color may be changed from a white background to green by using SETUP TG. Switch back to white by SETUP TW. See SETUP TERMINAL for more info on these and other options.

Other features of the display package include direct cursor addressing, clear end-of-line and end-of-screen, home, cancel line, and variable scroll rates. See Appendix E page 3.

The control-G bell is adjustable in pitch and duration. For information on the control codes for various screen operations, see Appendix E.

A Note about Motor Shutoff

There is no means to automatically shut off the drive motors in the Color Computer while running FLEX. Automatic motor shutoff is provided only while a program is waiting for keyboard input. If no key is pressed for a certain time, the motors are shut off. This time is adjustable using the SETUP command. The shutoff occurs after the cursor has blinked a certain number of times, thus the delay is also changed by changing the blink frequency. Note that virtual cursor blinks occur even if a non-blinking cursor is selected.

Printer Support

The P command (used to redirect output from a program to the printer) has been included as a memory-resident command. It redirects output to the Color Computer RS-232 port. Several options, adjustable using the SETUP command, allows almost any ASCII printer to be used. The baud rate is adjustable from 110 to 9600. The number of stop bits may be changed from 1 to 255. The printer may be either an auto-linefeed printer or a normal CR-LF printer.

Other FLEX Features

You may like to make some of these options permanant installations in your FLEX system. Of course, you could put the required SETUP command in your STARTUP file, but this uses up some time during bootup. A better way is to include the options right in the FLEX.SYS file. To do this you must create a temporary file called OPTIONS.BIN perhaps, to contain the options. to do this, type "SETUP FOPTIONS" followed by your options. Instead of affecting the options in memory, SETUP places them in the file. Now append FLEX.SYS to OPTIONS, creating a new file which will become your new FLEX.SYS file once renamed. On a single-drive system:

```
"APPEND FLEX.SYS OPTIONS.BIN NEWFLEX.SYS"
"DELETE FLEX.SYS"
"RENAME NEWFLEX.SYS FLEX.SYS"
"LINK FLEX.SYS"
```

On a two-drive system:

```
"SETUP F1.OPTIONS ..."
"APPEND 0.FLEX.SYS 1.OPTIONS.BIN 1.FLEX.SYS"
"LINK 1.FLEX.SYS"
```

Hooking up a Printer

You can attach a printer to your Color Computer and run it under FLEX very easily. The hookup is the same as it is for using the printer under COLOR BASIC. Serial data will come out the RS-232 Data Out pin, as long as the Data In pin is held high.

Use the SETUP command to tell FLEX what kind of printer you have. Set the baud rate by typing "SETUP PB1200" for a 1200 baud printer as an example. If the printer does not perform automatic linefeeds for each carriage return, type "SETUP PN". You may want to add more stop-bits. Typing "SETUP PS3" will give you 3. These three commands may be combined by typing "SETUP PNS3,B1200".

On some printers it may be necessary to adjust the baud rate count by +/- six percent to achieve reliable transmission. FLEX generates the best approximation of the baud rate you type that it can.

You may use the printer in two different ways. If you type a CTRL-P, everything that goes to the screen will also go to the printer until another CTRL-P is sent. Also, you may use the P command. This will redirect the output from a command to the printer. Try "P CAT".

FLEX Tutorial

What follows is a step by step tutorial which should help those new to the FLEX operating system with making backup copies of their FLEX disk and in understanding FLEX in general. Once through the Tutorial, read on to the more detailed information about FLEX features and commands, and see what applications you can come up with. Good luck and have fun!

FHL Color FLEX Tutorial

First you "Got Started with Color BASIC"
Then you "Went Ahead With Extended Color BASIC"
Now, **"Keep On Going With FLEX!"**

What exactly is "FLEX" and what will it let me do?

FLEX is a Disk Operating System (DOS) which allows your software to interact with your hardware. You may have a computer, and you may have a program written on a disk, but without a disk operating system you can put your disk in the drive and wait forever--the computer won't know the difference!! FLEX is like a bridge between the two which lets your program be read from its disk and be loaded into the computer's memory. NOW you can use it.

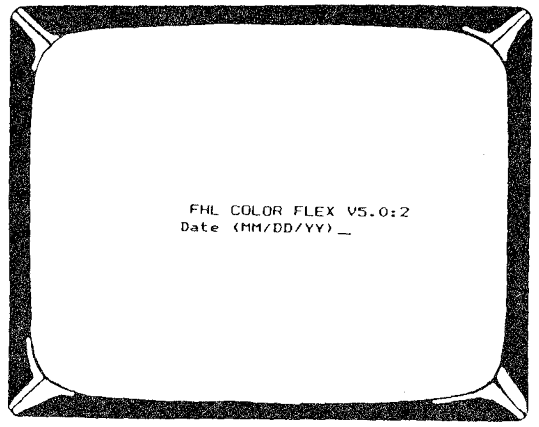
Now that I have my FLEX disk, what should I do with it?

How about trying to use it? Put it in the drive with the little white "write protect" sticker on top. Close the drive door and type:

RUN"FLEX ENTER

(When you see this symbol: ENTER That means you are to hit the enter key on your computer. Also, for simplicities sake, all information which you are to actually type in will appear in **BOLDFACE** type.)

Your drive motor should start to hum, in about six seconds the screen will appear like this:

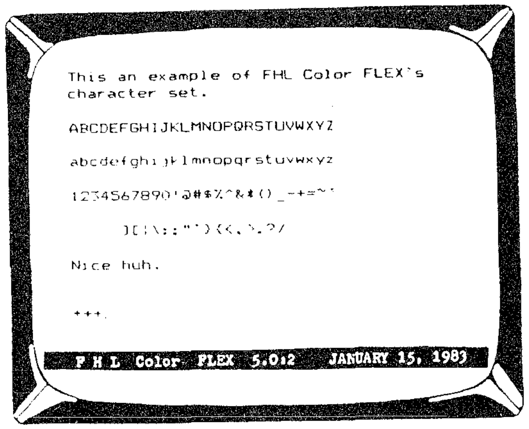


```
FHL COLOR FLEX V5.0:2
Date (MM/DD/YY) _
```

This is asking you for the date. Type in the numbers of the date in the same form. (MM/DD/YY)—don't include the parentheses!! For example:

2/15/83 (ENTER)

Not too tough right? Now FLEX will come back with:



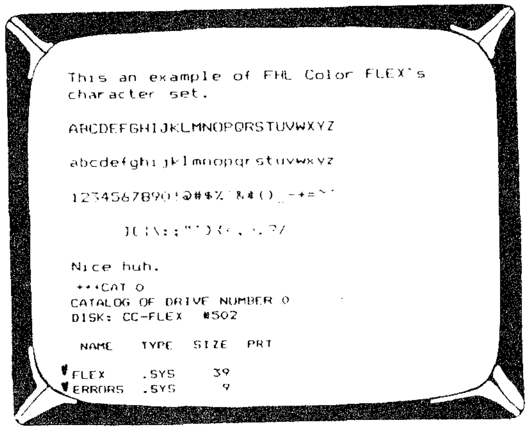
```
This an example of FHL Color FLEX's
character set.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

1234567890!@#$%^&*()_-+=~`

     ][|\:;"'}{<,>.?/

Nice huh.


+++_
F H L Color FLEX 5.0:2     JANUARY 15, 1983
```

+++_ means it's ready and waiting for you to tell it what to do. Well, let's find out what is on the disk. Type in:

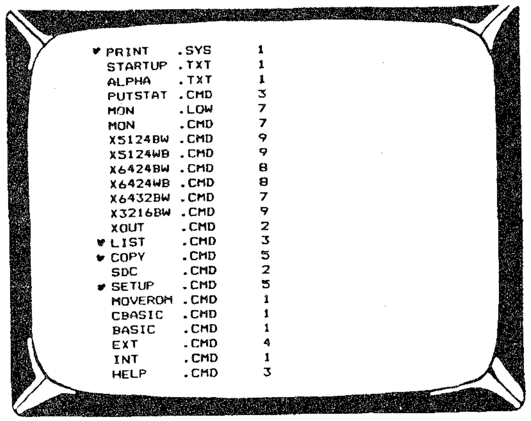
CAT 0 (ENTER)

(this is the number zero, not the letter O).

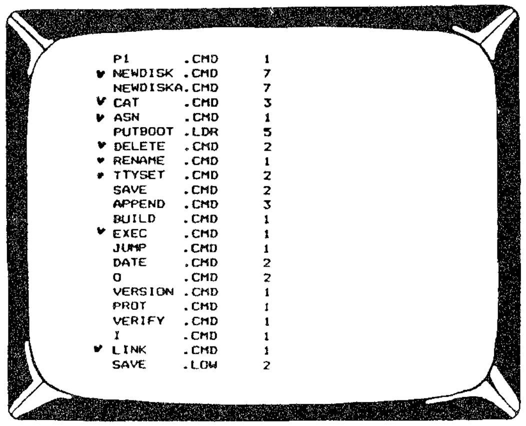
You have just "commanded" FLEX to give you a catalog of all the files that reside on the disk in drive #0 (which is the FLEX disk of course!). So now, what you should see on your screen is this:



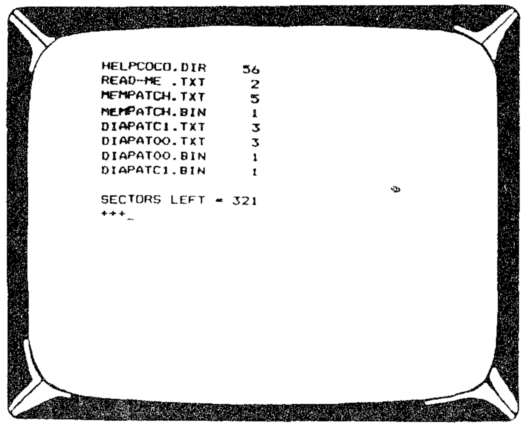
```
This an example of FHL Color FLEX's
character set.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

1234567890!@#$%^&*()_-+=~`

     ][|\:;"'}{<,>.?/

Nice huh.
+++CAT 0
CATALOG OF DRIVE NUMBER 0
DISK: CC-FLEX  #502

 NAME    TYPE  SIZE  PRT

♥FLEX     .SYS    39
♥ERRORS   .SYS     9
```



```
♥ PRINT    .SYS     1
  STARTUP  .TXT     1
  ALPHA    .TXT     1
  PUTSTAT  .CMD     3
  MON      .LOW     7
  MON      .CMD     7
  X5124BW  .CMD     9
  X5124WB  .CMD     9
  X6424BW  .CMD     8
  X6424WB  .CMD     8
  X6432BW  .CMD     7
  X3216BW  .CMD     9
  XOUT     .CMD     2
♥ LIST     .CMD     3
♥ COPY     .CMD     5
  SDC      .CMD     2
♥ SETUP    .CMD     5
  MOVEROM  .CMD     1
  CBASIC   .CMD     1
  BASIC    .CMD     1
  EXT      .CMD     4
  INT      .CMD     1
  HELP     .CMD     3
```



```
  P1       .CMD     1
♥ NEWDISK  .CMD     7
  NEWDISKA.CMD      7
♥ CAT      .CMD     3
♥ ASN      .CMD     1
  PUTBOOT  .LDR     5
♥ DELETE   .CMD     2
♥ RENAME   .CMD     1
♥ TTYSET   .CMD     2
  SAVE     .CMD     2
  APPEND   .CMD     3
  BUILD    .CMD     1
♥ EXEC     .CMD     1
  JUMP     .CMD     1
  DATE     .CMD     2
  O        .CMD     2
  VERSION  .CMD     1
  PROT     .CMD     1
  VERIFY   .CMD     1
  I        .CMD     1
♥ LINK     .CMD     1
  SAVE     .LOW     2
```



```
HELPCOCO.DIR    56
READ-ME .TXT     2
MEMPATCH.TXT     5
MEMPATCH.BIN     1
DIAPATC1.TXT     3
DIAPATOO.TXT     3
DIAPATOO.BIN     1
DIAPATC1.BIN     1

SECTORS LEFT = 321
+++_
```

Very Good!! You have now used FLEX.

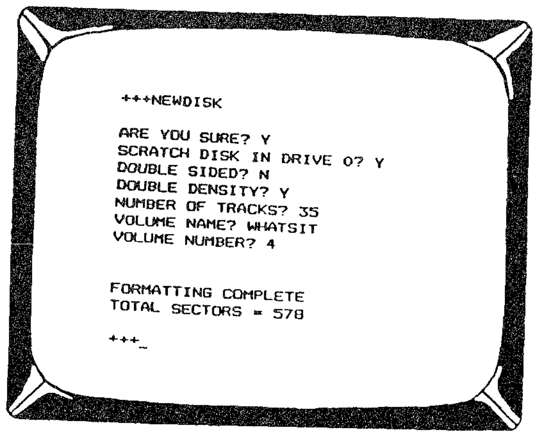
At this point, there are three items that should be clarified. First of all, what is a "write protect" sticker you may be wondering. Well, the name is relatively self explanatory. It "protects" the disk from being "written" on. Therefore, if you want to change anything already written on the disk, or if you want to save some information (write on the disk) you must "unprotect" it—in other words, take the little sticker off! Secondly, anytime you see a 0 used in a command, or to designate a drive, it is the number zero, not the letter O. Last, but not least, What's a drive #0 (or 1, 2, or 3 for that matter)? Well, your drives are numbered, which makes a very convenient way to tell FLEX which drive you want it to look in. If you only have one drive, then that's your drive #0 and FLEX will automatically look there for any information. With more than one drive, you have to tell FLEX which drive contains the information you want it to use. What's nice about two drives is that you can assign one to be your "system" drive, and the other to be your "working" drive. Your system drive contains your system disk—FLEX. Once you type RUN"FLEX" part of the information on your FLEX disk is loaded right into your computer's memory. This is the information that gets the whole system running. The rest of the information on your FLEX disk can be thought of as a "Dictionary" of sorts. When you tell FLEX to do something (i.e. type a command in) it will go to the system drive and "look up" on the disk what that particular command means it's supposed to do. The "work" drive contains the files of information that FLEX is supposed to "do something" to.

With a one drive system, your system and work drive are one in the same--called drive #0. This can be a bit of a pain in the neck at times if you want to command FLEX to do something to a file on a different disk.

Okay, now that you're just about a Pro, the next thing you'll probably want to do is to make a back-up copy of your FLEX disk.

There are five steps to this procedure. Again, there are FIVE steps to this procdure, so count them as you go. For simplicity's sake, we'll be assuming this is being done on a single drive system.

1. Prepare a new disk to have information put on it. This is called formatting, and is done using the NEWDISK command (if you have a double sided drive, you'll be using NEWDISKA instead). Here's an example of what the screen would look like during a typical formatting procedure:



```
+++NEWDISK

ARE YOU SURE? Y
SCRATCH DISK IN DRIVE 0? Y
DOUBLE SIDED? N
DOUBLE DENSITY? Y
NUMBER OF TRACKS? 35
VOLUME NAME? WHATSIT
VOLUME NUMBER? 4

FORMATTING COMPLETE
TOTAL SECTORS = 578

+++_
```

And here's how to get your screen to look that way.

Put your FLEX disk in the drive and after "booting up FLEX" (in other words after typing RUN"FLEX" etc.) type:

NEWDISK (ENTER)

FLEX will then come back with:

ARE YOU SURE?

Yes you are, so type: **Y**

Then FLEX will ask you if you want to:

SCRATCH DISK IN DRIVE 0?

BEFORE typing Y, remove your FLEX system disk and put in a new disk with no write protect sticker on it.

Now type: **Y**

FLEX will then ask the following questions:

DOUBLE-SIDED? Do you have a double sided drive? If not, type **N.** (If you do then you figure it out.)

DOUBLE DENSITY? You have a double density drive or you wouldn't have gotten this far, so type **Y.**

NUMBER OF TRACKS? That depends on the number of tracks your drive will support--on one side. Therefore, if you have a double sided drive, DO NOT multiply the number of tracks your drive has by two. Also realize that even though a 35 track disk may be read in a 40 track drive, a 40 track disk cannot be read in a 35 track (for example a Radio Shack) drive. So if you are formatting a disk that may be used in assorted drives, keep this fact in mind. One last note, neither 35 or 40 track disks may be read in an 80 track drive. (ENTER)

VOLUME NAME? Be original, why not call it DISK? (ENTER)

VOLUME NUMBER? This ones up to you.

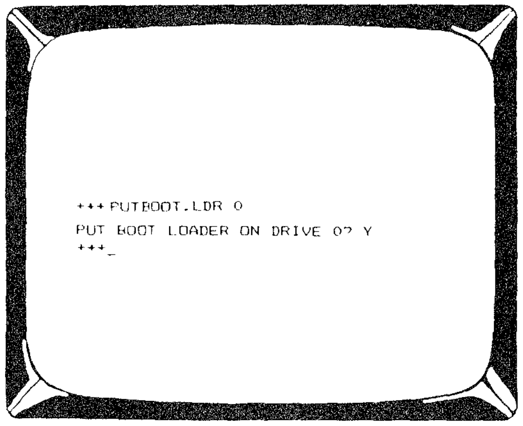
Now hit return and wait. The drive motor will begin running. This is an indication to you that formatting is taking place.

NOTE: Prior to the "FORMATTING COMPLETE" statement, you may see the message:

TRIMMING TRACK SIZE DOUBLE D.

This is not a problem--for more information on this, refer to the NEWDISK section of the FLEX manual.

2. The next step is to use PUTBOOT.LDR on your newly formatted disk. Here's what you'll see on your screen after performing this procedure:



```
+++PUTBOOT.LDR 0
PUT BOOT LOADER ON DRIVE 0? Y
+++_
```

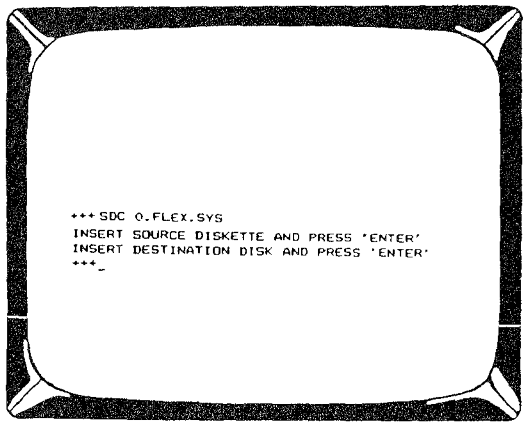
Now here's how you do it:

Put your FLEX disk back in the drive and type:

PUTBOOT.LDR 0 (the 0 tells FLEX which drive to go to). (ENTER)

PUT BOOT LOADER ON DRIVE 0? Now, you want to put the boot loader on your new disk, so this is your chance to switch disks—then type: **Y**

3. When you previously did a catalog (CAT 0) of your system disk, FLEX should have come back with a list of all the files it contained. In this step you're going to copy one of those files; the FLEX.SYS file. Here's the screen:



```
+++SDC 0.FLEX.SYS
INSERT SOURCE DISKETTE AND PRESS 'ENTER'
INSERT DESTINATION DISK AND PRESS 'ENTER'
+++_
```

And here's what you do:

Put your FLEX disk back in and type:

SDC 0.FLEX.SYS (ENTER)

Then when FLEX asks you to:

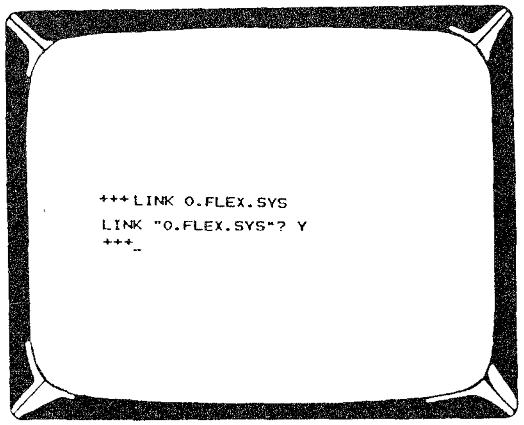
INSERT THE SOURCE DISKETTE AND PRESS 'ENTER'

Just hit enter, since the disk which contains the file you want to copy on it (your system disk) is already there. The destination disk is where you want the file to be copied to, so when FLEX asks you to:

INSERT DESTINATION DISK AND PRESS 'ENTER'

Put your new disk in and hit enter.

4. Three steps down, two to go. Now you've got to LINK the boot loader to the FLEX.SYS. Guess what command you're going to use? That's right, none other than the LINK command. Here's what to look forward to from your screen:



```
+++LINK 0.FLEX.SYS
LINK "0.FLEX.SYS"? Y
+++_
```

And this is how it's done:

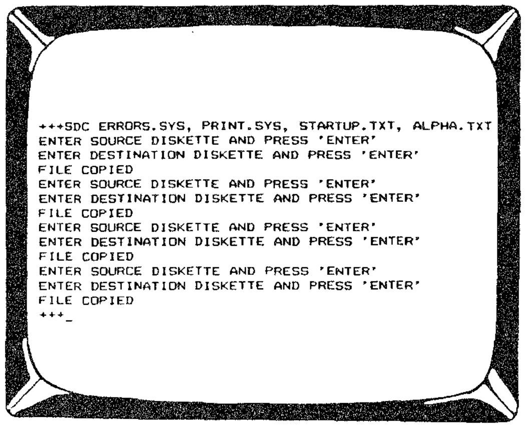
Put your FLEX disk in and type:

LINK 0.FLEX.SYS (ENTER)

When FLEX asks for confirmation, that's your cue to switch disks. Then type: Y

5. The only thing left for you to do at this point is to copy the individual files (remember all the ones that were listed out when you did a "CATalog" of your disk?) from your original FLEX disk to your new disk. This is done with the SDC command for a single drive system, and with the COPY command if you have more than one drive. With SDC you must write out each individual command file name that you are copying, although you may list more than one file in the same command line.

Here's an example of what you should see when you try to copy first four files from your original FLEX system disk using SDC:



```
+++SDC ERRORS.SYS, PRINT.SYS, STARTUP.TXT, ALPHA.TXT
ENTER SOURCE DISKETTE AND PRESS 'ENTER'
ENTER DESTINATION DISKETTE AND PRESS 'ENTER'
FILE COPIED
ENTER SOURCE DISKETTE AND PRESS 'ENTER'
ENTER DESTINATION DISKETTE AND PRESS 'ENTER'
FILE COPIED
ENTER SOURCE DISKETTE AND PRESS 'ENTER'
ENTER DESTINATION DISKETTE AND PRESS 'ENTER'
FILE COPIED
ENTER SOURCE DISKETTE AND PRESS 'ENTER'
ENTER DESTINATION DISKETTE AND PRESS 'ENTER'
FILE COPIED
+++_
```

And here's the necessary steps:

While your original FLEX disk is in the drive, type:

SDC ERRORS.SYS, PRINT.SYS, STARTUP.TXT, ALPHA.TXT (ENTER)

Now FLEX will come back with:

ENTER SOURCE DISKETTE AND PRESS 'ENTER'

Hit enter.

ENTER DESTINATION DISKETTE AND PRESS 'ENTER'

Switch disks and hit enter.

FILE COPIED

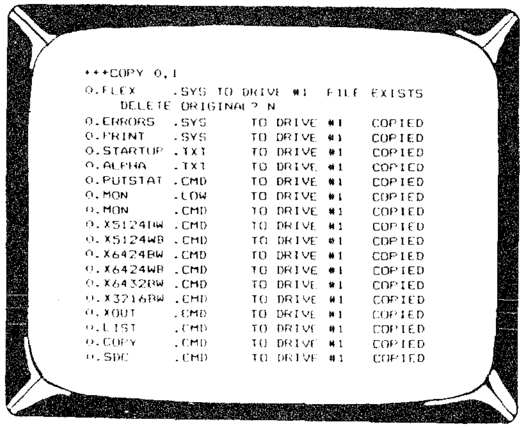
Then continue to switch the disks back and forth as FLEX repeats the above procedure for the next file, and the next file and finally the last file. Now look for that friendly little prompt;

+++_

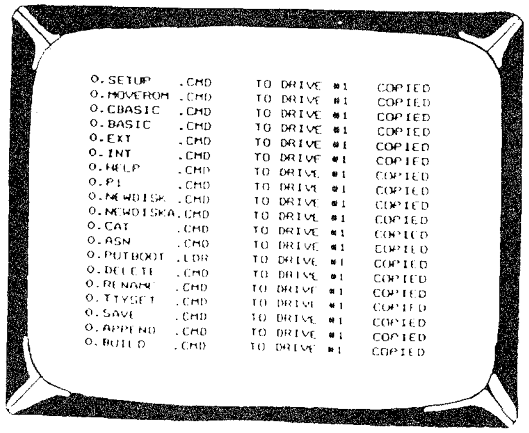
NOTE: Do not attempt to copy PUTBOOT.LDR using the SDC command because the results will not be of any use. Yes, that's right—you can't make a copy from a copy.

ALSO NOTE: Even though you may enter more than one file at a time in your initial command line, FLEX will only copy the files one at a time. So in the example above, this means that you would have to switch back and forth between your FLEX system disk and your new disk a total of four times.

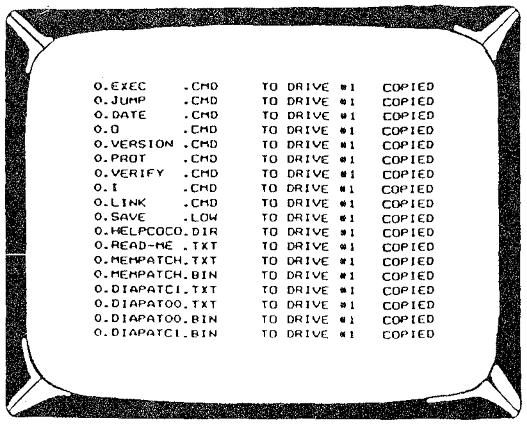
Okay all you "Two Drive Users," here's what you'll see on your screen:



```
+++COPY 0,1
0.FLEX     .SYS TO DRIVE #1  FILE EXISTS
   DELETE ORIGINAL? N
0.ERRORS   .SYS     TO DRIVE #1   COPIED
0.PRINT    .SYS     TO DRIVE #1   COPIED
0.STARTUP  .TXT     TO DRIVE #1   COPIED
0.ALPHA    .TXT     TO DRIVE #1   COPIED
0.PUTSTAT  .CMD     TO DRIVE #1   COPIED
0.MON      .LOW     TO DRIVE #1   COPIED
0.MON      .CMD     TO DRIVE #1   COPIED
0.X5124BW  .CMD     TO DRIVE #1   COPIED
0.X5124WB  .CMD     TO DRIVE #1   COPIED
0.X6424BW  .CMD     TO DRIVE #1   COPIED
0.X6424WB  .CMD     TO DRIVE #1   COPIED
0.X6432BW  .CMD     TO DRIVE #1   COPIED
0.X3216BW  .CMD     TO DRIVE #1   COPIED
0.XOUT     .CMD     TO DRIVE #1   COPIED
0.LIST     .CMD     TO DRIVE #1   COPIED
0.COPY     .CMD     TO DRIVE #1   COPIED
0.SDC      .CMD     TO DRIVE #1   COPIED
```



```
0.SETUP    .CMD     TO DRIVE #1   COPIED
0.MOVEROM  .CMD     TO DRIVE #1   COPIED
0.CBASIC   .CMD     TO DRIVE #1   COPIED
0.BASIC    .CMD     TO DRIVE #1   COPIED
0.EXT      .CMD     TO DRIVE #1   COPIED
0.INT      .CMD     TO DRIVE #1   COPIED
0.HELP     .CMD     TO DRIVE #1   COPIED
0.P1       .CMD     TO DRIVE #1   COPIED
0.NEWDISK  .CMD     TO DRIVE #1   COPIED
0.NEWDISKA.CMD      TO DRIVE #1   COPIED
0.CAT      .CMD     TO DRIVE #1   COPIED
0.ASN      .CMD     TO DRIVE #1   COPIED
0.PUTBOOT  .LDR     TO DRIVE #1   COPIED
0.DELETE   .CMD     TO DRIVE #1   COPIED
0.RENAME   .CMD     TO DRIVE #1   COPIED
0.TTYSET   .CMD     TO DRIVE #1   COPIED
0.SAVE     .CMD     TO DRIVE #1   COPIED
0.APPEND   .CMD     TO DRIVE #1   COPIED
0.BUILD    .CMD     TO DRIVE #1   COPIED
```



```
0.EXEC     .CMD     TO DRIVE #1   COPIED
0.JUMP     .CMD     TO DRIVE #1   COPIED
0.DATE     .CMD     TO DRIVE #1   COPIED
0.O        .CMD     TO DRIVE #1   COPIED
0.VERSION  .CMD     TO DRIVE #1   COPIED
0.PROT     .CMD     TO DRIVE #1   COPIED
0.VERIFY   .CMD     TO DRIVE #1   COPIED
0.I        .CMD     TO DRIVE #1   COPIED
0.LINK     .CMD     TO DRIVE #1   COPIED
0.SAVE     .LOW     TO DRIVE #1   COPIED
0.HELPCOCO.DIR      TO DRIVE #1   COPIED
0.READ-ME  .TXT     TO DRIVE #1   COPIED
0.MEMPATCH.TXT      TO DRIVE #1   COPIED
0.MEMPATCH.BIN      TO DRIVE #1   COPIED
0.DIAPATC1.TXT      TO DRIVE #1   COPIED
0.DIAPATOO.TXT      TO DRIVE #1   COPIED
0.DIAPATOO.BIN      TO DRIVE #1   COPIED
0.DIAPATC1.BIN      TO DRIVE #1   COPIED
```

This is all you have to do:

Put your FLEX system disk in drive 0 and your new disk in drive 1 and type:

COPY 0,1 (ENTER)

Voila! You're all done! Now, put your master disk away in a safe place, your new disk in the drive, your FLEX manual on your lap, and learn how to use the darn thing!!

Let's say you just bought some new software to run under FLEX (in other words, is "FLEX compatible"). What should you do with it first?

Just to see what you actually got, do a CAT (catalog) of your new disk. Now, realize that CAT is a FLEX command and is only found on your FLEX disk (this is what is meant by "Disk Resident Commands"). So, if you have a single disk drive system and put your new disk in and type CAT, the system will look on the disk to see what CAT means, and guess what? It won't be there, and your computer will come back with the message NOT FOUND. Therefore you must copy the CAT command onto your new disk. This is quite simple and is done by using the FLEX SDC command. Put your FLEX disk in, type SDC CAT.CMD then hit enter. When asked for the source diskette, hit enter. When asked for the destination diskette, take the write protect sticker off the new disk and insert it in the drive. Hit enter. Remove the disk after the copy is complete and put the write protect sticker back on and then just do a catalog of the disk (CAT 0). Now you at least know what it is you spent your money on, and you can then make a copy or find out what the various files do as explained in the documentation that accompanies your new software.

FHL Color FLEX

Disk Operating System

GENERAL INFORMATION

THE FLEX DISK OPERATING SYSTEM

I. INTRODUCTION

The FLEX™ Operating System is a very versatile and flexible operating system. It provides the user with a powerful set of system commands to control all disk operations directly from the user's terminal. The systems programmer will be delighted with the wide variety of disk access and file management routines available for personal use. Overall, FLEX is one of the most powerful operating systems available today.

The FLEX Operating System is comprised of three parts, the File Management System (FMS), the Disk Operating System (DOS), and the Utility Command Set (UCS). Part of the power of the overall system lies in the fact that the system can be greatly expanded by simply adding additional utility commands. The user should expect to see many more utilities available for FLEX in the future. Some of the other important features include: fully dynamic file space allocation, the automatic "removal" of defective sectors from the disk, automatic space compression and expansion on all text files, complete user environment control using the TTYSET utility command, and uniform disk wear due to the high performance dynamic space allocator.

The UCS currently contains many very useful commands. These programs reside on the system disk and are only loaded into memory when needed. This means that the set of commands can be easily extended at any time, without the necessity of replacing the entire operating system. The utilities provided with FLEX perform such tasks as the saving, loading, copying, renaming, deleting, appending, and listing of disk files. There is an extensive CATalog command for examining the disk's file directory. Several environment control commands are also provided. Overall, FLEX provides all of the necessary tools for the user's interaction with the disk.

----
* FLEX is a registered trademark of Technical Systems Consultants, Inc.

II. SYSTEM REQUIREMENTS

FLEX requires random access memory from location 0000 through location 2FFF hex (12K). Memory is also required from C000 (48K) through DFFF hex (56K), where the actual operating system resides. The system also assumes at least 2 disk drives are connected to the controller and that they are configured as drives #0 and #1. You should consult the disk drive instructions for this information. FLEX interfaces with the disk controller through a section of driver routines and with the operator console or terminal through a section of terminal I/O routines.

III. GETTING THE SYSTEM STARTED

Each FLEX system diskette contains a binary loader for loading the operating system into RAM. There needs to be some way of getting the loader off of the disk so it can do its work. This can be done by either hand entering the bootstrap loader provided with the disk system, or by using the boot provided in ROM if appropriate to FLEX.

As a specific example, suppose the system we are using has SWTPc's S-BUG installed and we wish to run FLEX. The first step is to power on all equipment and make sure the S-BUG prompt is present (>). Next insert the system diskette into drive 0 (the boot must be performed with the disk in drive 0) and close the door on the drive. Type "D" on the terminal if using a full size floppy system or "U" if a minifloppy system. The disk motors should start, and after about 2 seconds, the following should be displayed on the terminal:

```
FLEX X.X
DATE (MM,DD,YY)?

+++
```

The name FLEX identifies the operating system and the X.X will be the version number of the operating system. At this time the current date should be entered, such as 7,3,79. The FLEX prompt is the three plus signs (+++), and will always be present when the system is ready to accept an operator command. The '+++' should become a familiar sight and signifies that FLEX is ready to work for you!

IV. DISK FILES AND THEIR NAMES

All disk files are stored in the form of 'sectors' on the disk and in this version, each sector contains 256 'bytes' of information. Each byte can contain one character of text or one byte of binary machine information. A maximum of 340 user-accessible sectors will fit on a single-sided mini disk or 1140 sectors on a single-sided full size floppy. Double-sided disks would hold exactly twice that number of sectors. Double-density systems will hold more still. The user, however, need not keep count, for the system does this automatically. A file will always be at least one sector long and can have as many as the maximum number of sectors on the disk. The user should not be concerned with the actual placement of the files on the disk since this is done by the operating system. File deletion is also supported and all previously used sectors become immediately available again after a file has been deleted.

All files on the disk have a name. Names such as the following are typical:

```
PAYROLL
INVNTORY
TEST1234
APRIL-78
WKLY-PAY
```

Anytime a file is created, referenced, or deleted, its name must be used. Names can be most anything but must begin with a letter (not numbers or symbols) and be followed by at most 7 additional characters, called 'name characters'. These 'name characters' can be any combination of the letters 'A' through 'Z' or 'a' through 'z', any digit '0' through '9', or one of the two special characters, the hyphen (-) or the underscore '_', (a left arrow on some terminals).

File names must also contain an 'extension'. The file extension further defines the file and usually indicates the type of information contained therein. Examples of extensions are: TXT for text type files, BIN for machine readable binary encoded files, CMD for utility command files, and BAS for BASIC source programs. Extensions may contain up to 3 'name characters' with the first character being a letter. Most of the FLEX commands assume a default extension on the file name and the user need not be concerned with the actual extension on the file. The user may at anytime assign new extensions, overiding the default value, and treat the extension as just part of the file name. Some examples of file names with their extensions follow:

```
APPEND.CMD
LEDGER.BAS
TEST.BIN
```

Note that the extension is always separated from the name by a period '.'. The period is the name 'field separator'. It tells FLEX to treat the following characters as a new field in the name specification.

A file name can be further refined. The name and extension uniquely define a file on a particular drive, but the same name may exist on several drives simultaneously. To designate a particular drive a 'drive number' is added to the file specification. It consists of a single digit (0-3) and is separated from the name by the field separator '.'. The drive number may appear either before the name or after it (after the extension if it is given). If the drive is not specified, the system will default to either the 'system' drive or the 'working' drive. These terms will be described a little later.

Some examples of file specifications with drive numbers follow:

```
0.BASIC
MONDAY.2
1.TEST.BIN
LIST.CMD.1
```

In summary, a file specification may contain up to three fields separated by the field separator. These fields are; 'drive', 'name', and 'extension'. The rules for the file specification can be stated quite concisely using the following notation:

```
[<drive>.]<name>[.<extension>]
  or
<name>[.<extension>][.<drive>]
```

The '<>' enclose a field and do not actually appear in the specification, and the '[]' surround optional items of the specification. The following are all syntactically correct:

```
0.NAME.EXT
NAME.EXT.0
NAME.EXT
0.NAME
NAME.0
NAME
```

Note that the only required field is the actual 'name' itself and the other values will usually default to predetermined values. Studying the above examples will clarify the notation used. The same notation will occur regularly throughout the manual.

V. ENTERING COMMANDS

When FLEX is displaying '+++', the system is ready to accept a command line. A command line is usually a name followed by certain parameters depending on the command being executed. There is no 'RUN' command in FLEX. The first file name on a command line is always loaded into memory and execution is attempted. If no extension is given with the file name, 'CMD' is the default. If an extension is specified, the one entered is the one used. Some examples of commands and how they would look on the terminal follow:

```
+++TTYSET
+++TTYSET.CMD
+++LOOKUP.BIN
```

The first two lines are identical to FLEX since the first would default to an extension of CMD. The third line would load the binary file 'LOOKUP.BIN' into memory and, assuming the file contained a transfer address, the program would be executed. A transfer address tells the program loader where to start the program executing after it has been loaded. If you try to load and execute a program in the above manner and no transfer address is present, the message, 'NO LINK' will be output to the terminal, where 'link' refers to the transfer address. Some other error messages which can occur are 'WHAT?' if an illegal file specification has been typed as the first part of a command line, and 'NOT THERE' if the file typed does not exist on the disk.

During the typing of a command line, the system simply accepts all characters until a 'RETURN' key is typed. Any time before typing the RETURN key, the user may use one of two special characters to correct any mistyped characters. One of these characters is the 'back space' and allows deletion of the previously typed character. Typing two back spaces will delete the previous two characters. The back space is initially defined to be a 'control H' but may be redefined by the user using the TTYSET utility command. The second special character is the line 'delete' character. Typing this character will effectively delete all of the characters which have been typed on the current line. A new prompt will be output to the terminal, but instead of the usual '+++' prompt, to show the action of the delete character, the prompt will be '???'. Any time the delete character is used, the new prompt will be '???', and signifies that the last line typed did not get entered into the computer. The delete character is initially a 'control X' but may also be redefined using TTYSET.

As mentioned earlier, the first name on a command line is always interpreted as a command. Following the command is an optional list of names and parameters, depending on the particular command being entered. The fields of a command line must be separated by either a space or a comma. The general format of a command line is:

```
<command>[,<list of names and parameters>]
```

A comma is shown, but a space may be used. FLEX also allows several commands to be entered on one command line by use of the 'end of line' character. This character is initially a colon (':'), but may be user defined with the TTYSET utility. By ending a command with the end of line character, it is possible to follow it immediately with another command. FLEX will execute all commands on the line before returning with the '+++' prompt. An error in any of the command entries will cause the system to terminate operation of that command line and return with the prompt. Some examples of valid command lines follow:

```
+++CAT 1
+++CAT 1:ASN S=1
+++LIST LIBRARY:CAT 1:CAT 0
```

As many commands may be typed in one command line as desired, but the total number of characters typed must not exceed 128. Any excess characters will be ignored by FLEX.

One last system feature to be described is the idea of 'system' and 'working' drives. As stated earlier, if a file specification does not specifically designate a drive number, it will assume a default value. This default value will either be the current 'system' drive assignment or the current 'working' drive assignment. The system drive is the default for all command names, or in other words, all file names which are typed first on a command line. Any other file name on the command line will default to the working drive. This version of FLEX also supports automatic drive searching. When in the auto search mode if no drive numbers are specified, the operating system will first search drive 0 for the file. If the file is not found, drive 1 will be searched and so on. When the system is first initialized the auto drive searching mode will be selected. At this time, all drive defaults will be to drive 0. It is sometimes convenient to assign drive 1 as the working drive in which case all file references, except commands, will automatically look on drive 1. It is then convenient to have a diskette in drive 0 with all the system utility commands on it (the 'system drive'), and a disk with the files being worked on in drive 1 (the 'working drive'). If the system drive is 0 and the working drive is 1, and the command line was:

```
+++LIST TEXTFILE
```

FLEX would go to drive 0 for the command LIST and to drive 1 for the file TEXTFILE. The actual assignment of drives is performed by the ASN utility. See its description for details.

VI. COMMAND DESCRIPTIONS

There are two types of commands in FLEX, memory resident (those which actually are part of the operating system) and disk utility commands (those commands which reside on the disk and are part of the UCS).

There are four memory resident commands included in FHL Color FLEX: GET, MON, ROM, and P. Even though they are not considered part of the Disk Utility Command Set, for simplicities sake they will be discussed in the UCS section of this manual.

GENERAL SYSTEM INFORMATION

I. DISK CAPACITY

Each sector of a FLEX disk contains 252 characters or bytes of user data (4 bytes of 256 byte sector are used by the system). Thus a single-sided mini disk has 340 sectors or 85,680 characters or bytes of user information. A single-sided full size disk has 1140 sectors or 287,280 bytes of user data. Double-sided disks would contain exactly twice these amounts.

II. WRITE PROTECT

Floppy disks can usually be physically write protected to prevent FLEX from performing a write operation. Any attempt to write to such a disk will cause an error message to be issued. It is good practice to write protect disks which have important files on them.

A mini disk can be write protected by placing a piece of opaque tape over the small rectangular cutout on the edge of the disk. Full size floppys are just the opposite. In order to write protect a full size disk, you must remove the tape from the cutout. In other words, the notch must be exposed to write protect the disk. Some full size disks do not have this cutout, and therefore cannot be write protected.

III. THE 'RESET' BUTTON

The RESET button on the back panel of your computer should NEVER BE PRESSED DURING A DISK OPERATION. There should never be a need to reset the machine while in FLEX. If the machine is reset and the system is writing data on the disk, it is possible that the entire disk will become damaged. Again, never press 'reset' while the drive is operating! Refer to the 'escape' character in TTYSET for ways of stopping FLEX.

IV. NOTES ON THE 'P' COMMAND

On a standard FLEX system, the 'P' command tries to load a printer driver file named 'PRINT.SYS' from the same disk which P itself was retrieved (after first making sure that a driver has not already been loaded in). FHL Color FLEX utilizes P in a different manner:

P is a memory resident command which simply redirects I/O to the printer port of the Color Computer. Instead of playing the part of the hotel doorman (the function of the standard P) calling a taxi (PRINT.SYS) for the waiting patron (the file to be output), P in this case takes the patron directly to a location where he knows the "hotel taxi" is waiting. This particular hotel is not serviced by any other taxis, so the doorman (P) knows there is only one way out - the hotel taxi.

As presently is and has been the case on the Color Computer, there is only one output port - the RS 232 port. Similar to the hotel with the "built-in" taxi, FHL Color FLEX has a built in (memory-resident) printer driver just sitting there at the RS232 port waiting for P to send it some data to be printed. The reason it can be built in (memory-resident) is because there is only need for one standard driver since all data on the Color Computer is being sent out one standard port (the RS 232).

For more information on the standard FLEX PRINT.SYS file, see page 53 of the 'Advanced Programmer's Guide.'

VI. SYSTEM ERROR NUMBERS

Any time that FLEX detects an error during an operation, an appropriate error message will be displayed on the terminal. FLEX internally translates a derived error number into a plain language statement using a look-up table called ERROR.SYS. If you have forgotten to copy this .SYS file onto a disk that you are using, FLEX will report a corresponding number as shown below:

```
DISK ERROR  #xx
```

where 'xx' is a decimal error number. The table below is a list of these numbers and what error they represent.

ERROR #	MEANING
1	ILLEGAL FMA FUNCTION CODE ENCOUNTERED
2	THE REQUESTED FILE IS IN USE
3	THE FILE SPECIFIED ALREADY EXISTS
4	THE SPECIFIED FILE COULD NOT BE FOUND
5	SYSTEM DIRECTORY ERROR-REBOOT SYSTEM
6	THE SYSTEM DIRECTORY IS FULL
7	ALL AVAILABLE DISK SPACE HAS BEEN USED
8	READ PAST END OF FILE
9	DISK FILE READ ERROR
10	DISK FILE WRITE ERROR
11	THE FILE OR DISK IS WRITE PROTECTED
12	THE FILE IS PROTECTED-FILE NOT DELETED
13	ILLEGAL FILE CONTROL BLOCK SPECIFIED
14	ILLEGAL DISK ADDRESS ENCOUNTERED
15	AN ILLEGAL DRIVE NUMBER WAS SPECIFIED
16	DRIVE NOT READY
17	THE FILE IS PROTECTED-ACCESS DENIED
18	SYSTEM FILE STATUS ERROR
19	FMS DATA INDEX RANGE ERROR
20	FMS INACTIVE-REBOOT SYSTEM
21	ILLEGAL FILE SPECIFICATION
22	SYSTEM FILE CLOSE ERROR
23	SECTOR MAP OVERFLOW-DISK TOO SEGMENTED
24	NON-EXISTENT RECORD NUMBER SPECIFIED
25	RECORD NUMBER MATCH ERROR-FILE DAMAGED
26	COMMAND SYNTAX ERROR-RE-TYPE COMMAND
27	THAT COMMAND IS NOT ALLOWED WHILE PRINTING
28	WRONG HARDWARE CONFIGURATION

For more details concerning the meanings of these error messages, consult the 'Advanced Programmer's Guide' (pgs. 38 - 40).

VII. FLEX OPERATING SYSTEM INPUT/OUTPUT SUBROUTINES

In order for the FLEX I/O functions to operate properly, all user program character input/output subroutines should be vectored thru the FLEX operating system rather than the computer's monitor. Below is a list of FLEX's I/O subroutines and a brief description of each. All given addresses are in hexadecimal.

GETCHR at $CD15
This subroutine is functionally equivalent to S-BUG's character input routine. This routine will look for one character from the control terminal (I/O port #1) and store it in the A accumulator. Once called, the input routine will loop within itself until a character has been input. Anytime input is desired, the call JSR GETCHR or JSR $CD15 should be used.

GETCHR automatically sets the 8th bit to 0 and does not check for parity. A call to this subroutine affects the processor's registers as follows:

```
ACC. A   loaded with the character input from the terminal
B,X,Y,U  not affected
```

PUTCHR at $CD18
This subroutine is used to output one character from the computer to the control port (I/O port #1). It is functionally equivalant to the output character routine in S-BUG.

To use PUTCHR, the character to be output should be placed in the A accumulator in its ASCII form. For example, to output the letter 'A' on the control terminal, the following program should be used:

```
LDA    #$41
JSR    $CD18
```

The processor's registers are affected as follows:

```
ACC. A   changed internally
B,X,Y,U  not affected
```

PSTRNG at $CD1E
PSTRNG is a subroutine used to output a string of text on the control terminal. When address $CD1E is called, a carriage return and line feed will automatically be generated and data output will begin at the location pointed to by the index register. Output will continue until a hex 04 is seen. The same rules for using the ESCAPE and RETURN keys for stopping output apply as described earlier.

The accumulator and register status after using PSTRNG are as follows:

```
ACC. A   Changed during the operation
ACC. B   Unchanged
X        Contains the memory location of the last character read from the
           string (usually the 04 unless stopped by the ESC key)
Y,U      Unchanged
```

NOTE: The ability of using backspace and line delete characters is a function of your user program and not of the FLEX I/O routines described above.

For additional information consult the 'Advanced Programmer's Manual'.

STAT at $CD4E
This routine is used to determine the "status" of the input device. That is, to see if a character has been typed on the input terminal keyboard. Its function is to check for characters such as the ESCAPE key in FLEX which allows breaking of the output. This routine returns an EQual condition if no character was hit and a Not-Equal condition if a character was hit. No registers, except for the condition codes, may be altered.

VIII. BOOTING THE FLEX DISK OPERATING SYSTEM

In order to read FLEX from the system disk upon powering up your system, you must have a short program in RAM or ROM memory. This program is called a 'bootstrap' loader.

With FHL Color FLEX, typing in RUN"FLEX" will call the BASIC bootstrap loader called "FLEX".

Non-Color Computer Users should use the boot supplied with the hardware if compatible with FLEX. A sample boot (for the SWTPc mini system) is given here for reference.

If the system does not boot properly, re-position the system disk in the drive and re-execute the bootstrap loader.

```
0100 B6   E018   START   LDA    COMREG    TURN MOTOR ON
0103 86   00             LDA    #0
0105 B7   E014           STA    DRVREG
0108 8E   0000           LDX    #0000
010B 3D          OVR     MUL              DELAY FOR SPEED UP
010C 30   1F             LEAX   -1,X
010E 26   FB             BNE    OVR
0110 C6   0F             LDB    #$0F      RESTORE
0112 F7   E018           STB    COMREG
0115 8D   2B             BSR    RETURN
0117 F6   E018   LOOP1   LDB    COMREG
011A C5   01             BITB   #1
011C 26   F9             BNE    LOOP1
011E 86   01             LDA    #1
0120 B7   E01A           STA    SECREG
0123 8D   1D             BSR    RETURN
0125 C6   8C             LDB    #$8C      READ WITH LOAD
0127 F7   E018           STB    COMREG
012A 8D   16             BSR    RETURN
012C 8E   C000           LDX    #$C000
012F C5   02     LOOP2   BITB   #2        DRQ?
0131 27   05             BEQ    LOOP3
0133 B6   E01B           LDA    DATREG
0136 A7   80             STA    0,X+
0138 F6   E018   LOOP3   LDB    COMREG
013B C5   01             BITB   #1        BUSY?
013D 26   F0             BNE    LOOP2
013F 7E   C000           JMP    $C000
0142 8D   00     RETURN  BSR    RTN
0144 39          RTN     RTS
```

IX. REQUIREMENTS FOR THE 'PRINT.SYS' PRINTER DRIVER

(This does not apply to FHL Color FLEX. See pg. 1.9)

Standard TSC FLEX, as supplied, includes a printer driver that will work with most parallel type printers, such as the SWTPC PR-40. If desired, the printer driver may be changed to accomodate other types of printers. Included is the source listing for the supplied driver. Additional information on the requirements for the PRINT.SYS driver can be found in the 'Advanced Programmer's Guide' on page 53.

1) The driver must be in a file called PRINT.SYS

2) Three separate routines must be supplied, a printer initialization routine (PINIT at $CCC0), a check ready routine (PCHK at $CCD8), and an output character routine (POUT at $CCE4).

3) When the POUT routine is called by FLEX, the character to be output will be in the A accumulator. The output routine must not destroy the B, X, Y, or U registers. PINIT may destroy any registers. PCHK may NOT alter any registers.

4) The routines MUST start at the addresses specified, but may be continued anywhere in memory if there is not room where specified. If placed elsewhere in memory, be certain they do not conflict with any utilities or programs which will use them.

5) All three routines must end with a return from subroutine instruction (RTS).

```
                         *
                         * PRINT.SYS PIA DRIVERS FOR GENERAL CASE PRINTER
                         *

                  E01C   PIA      EQU     $E01C      PIA ADDRESS FOR PORT #7

                         *
                         * PRINTER INITIALIZATION (MUST BE AT $CCC0)
                         *
CCC0                              ORG     $CCC0      MUST RESIDE AT $CCC0
CCC0 86     3A           PINIT    LDA     #$3A       SELECT DATA DIRECTION REG.
CCC2 B7     E01D                  STA     PIA+1      BY WRITING 0 IN DDR CONTROL
CCC5 86     FF                    LDA     #$FF       SELECT ALL OUTPUT LINES
CCC7 B7     E01C                  STA     PIA        PUT IN DATA DIRECTION REG.
CCCA 86     3E                    LDA     #$3E       SET UP FOR TRANSITION CHECKS
CCCC B7     E01D                  STA     PIA+1      AND ENABLE OUTPUT REGISTER
CCCF 39                           RTS
                         * PRINTER READY ROUTINE
CCD0 7D     E01C         PREADY   TST     PIA        RESET PIA READY INDICATION
CCD3 73     CCE3                  COM     PFLAG      SET THE PRINTER READY FLAG
CCD6 39                           RTS
```

```
                   *
                   * CHECK FOR PRINTER READY (MUST BE AT $CCD8)
                   *
CCD8                        ORG    $CCD8      PRINT TEST AT $CCD8
CCD8 7D   CCE3     PCHK     TST    PFLAG      TEST FOR PRINTER READY
CCDB 2B   05                BMI    PCHKX      IF NEGATIVE, PRINTER READY
CCDD 7D   E01D              TST    PIA+1      CHECK FOR TRANSITION
CCE0 2B   EE                BMI    PREADY     IF MINUS, PRINTER NOW READY
CCE2 39            PCHKX    RTS
                   * PRINTER READY FLAG
CCE3 FF            PFLAG    FCB    $FF        PRINTER READY FLAG

                   *
                   * PRINTER OUTPUT CHARACTER ROUTINE (MUST BE AT $CCE4)
                   *
CCE4                        ORG    $CCE4      MUST RESIDE AT $CCE4
CCE4 8D   F2       POUT     BSR    PCHK       TEST FOR PRINTER READY
CCE6 2A   FC                BPL    POUT       LOOP UNTIL PRINTER READY
CCE8 7F   CCE3              CLR    PFLAG      SET PRINTER FLAG NOT READY
CCEB B7   E01C              STA    PIA        SET DATA IN OUTPUT REGISTER
CCEE 86   36                LDA    #$36       SET DATA READY, HIGH TO LOW
CCF0 8D   02                BSR    POUTB      STUFF BYTE INTO THE PIA
CCF2 86   3E                LDA    #$3E       THEN SEARCH FOR TRANSITION
CCF4 B7   E01D     POUTB    STA    PIA+1      OF LOW LEVEL TO HIGH LEVEL
CCF7 39                     RTS

                            END
```

SAMPLE DRIVERS FOR SERIAL PRINTER

The following listing is a sample set of drivers for a serial type printer using an ACIA as its interface. This would be of interest to Color Computer Users using some type of expansion device allowing access to a printer port other than the standard RS 232 port. This set of drivers is not supplied on disk. In order to use these drivers, you must type in the source and assemble it.

```
                        *
                        * PRINT.SYS DRIVERS FOR GENERAL SERIAL PRINTER
                        * CHANGE ACIA EQUATE IF NECESSARY
                        *

                  E01C  ACIA    EQU    $E01C     ACIA ADDRESS FOR PORT #7

                        *
                        * PRINTER INITIALIZATION (MUST BE AT $CCC0)
                        *
 CCC0                           ORG    $CCC0     MUST RESIDE AT $CCC0
 CCC0 86    13          PINIT   LDA    #$13      RESET ACIA
 CCC2 B7    E01C                STA    ACIA
 CCC5 86    11                  LDA    #$11      SET 8 BITS & 2 STOP
 CCC7 B7    E01C                STA    ACIA
 CCCA 39                        RTS              RETURN

                        *
                        * CHECK FOR PRINTER READY (MUST BE AT $CCD8)
                        *
 CCD8                           ORG    $CCD8     PRINT TEST AT $CCD8
 CCD8 34    04          PCHK    PSHS   B         SAVE B ACC.
 CCDA F6    E01C                LDB    ACIA      GET STATUS
 CCDD 56                        RORB             GET TDR BIT INTO
 CCDE 56                        RORB             SIGN POSITION
 CCDF 56                        RORB
 CCE0 35    04                  PULS   B         RESTORE B ACC.
 CCE2 39                        RTS              RETURN

                        *
                        * PRINTER OUTPUT CHARACTER ROUTINE (MUST BE AT $CCE4)
                        *
 CCE4                           ORG    $CCE4     MUST RESIDE AT $CCE4
 CCE4 34    04          POUT    PSHS   B         SAVE B ACC.
 CCE6 F6    E01C        POUT2   LDB    ACIA      GET STATUS
 CCE9 57                        ASRB             GET TDR BIT
 CCEA 57                        ASRB             INTO CARRY
 CCEB 24    F9                  BCC    POUT2     LOOP IF NOT READY
 CCED 35    04                  PULS   B         RESTORE B ACC.
 CCEF B7    E01D                STA    ACIA+1    WRITE OUT THE CHAR.
 CCF2 39                        RTS              RETURN

                                END
```

DISK COMPATIBILITY

Disks created under 6809 FLEX 9.0 are compatible with those created under 6800 FLEX 1.0 on the 8" drives or 6800 FLEX 2.0 on the 5" drives. The reverse is also true, meaning that FLEX 9.0 can read disks created by one of those 6800 FLEX systems. This means that transferring text files will require nothing more than copying with the COPY command. In fact it is not even necessary to put the files on a new disk. As long as a disk is being used for work files only (no disk command files) it may be used interchangeably.

The one place where the disks are different is in the bootstrap loader which the NEWDISK command places on track 0 when a disk is initialized. Obviously the loader must be different for 6800 and 6809. This simply means that a disk initialized with the 6809 NEWDISK command cannot be used to boot 6800 FLEX and vice versa.

The new double-density system is an exception to all the above. It cannot be used to read disks created by the original 6800 single-density system. Any disks, however, created as single-density with the new double-density version of NEWDISK (done by answering 'N' to the prompt 'Double-Sided Disk?') can be read on either a single or double density system. This is because the new double-density NEWDISK writes FF's in certain gap areas whereas the old single-density NEWDISK wrote 00's. The single-density controller board (which uses the Western Digital 1771) can read either type, but the double-density board (which uses the Western Digital 1791) can only read the type with FF's.

SOFTWARE COMPATIBILITY

6809 object code is NOT at all compatible with 6800 object code. This means you cannot run binary command files from a 6800 system on a 6809 system. Since 6809 FLEX can read a 6800 FLEX disk and vice versa, you must be careful not to execute a 6800 command in a 6809 system and again, vice versa.

Where the 6809 and 6800 ARE compatible is in the source code. Thus, if you have the source listing for a 6800 program on disk, it can be reassembled by the 6809 assembler to produce executable 6809 object code. Of course if the program calls any routines from FLEX, these addresses will have to be changed since 6809 FLEX resides at $C000 (6800 FLEX is at $A000). This is usually a matter of simply changing all occurrences of '$A' to '$C' and all '$B' to '$D' with the editor.

ADAPTING FLEX

The FLEX 9.0 disk supplied has two copies of the FLEX object code. One is called FLEX.SYS and is ready to boot up with SWTPc disk hardware. The second is called FLEX.COR which represents the CORe or main body of FLEX. It differs from the bootable form of FLEX in that it does not have any terminal or disk I/O routines built in. This allows the user to modify these I/O drivers, if desired, to produce a customized version of FLEX. Note that in order to produce this customized version you must have FLEX up and running so you will need the bootable version (FLEX.SYS). The customized terminal and disk I/O routines are supplied in two packages. We will discuss them separately and then examine how to add them onto FLEX.COR to produce a new, customized, bootable version of FLEX.

The CUSTOM I/O DRIVER PACKAGE

This package allows the user to alter the functioning of the terminal I/O and the functioning of printer spooling. Nine routines and two interrupt vectors are set up in this package. There is a space reserved for these routines beginning at location $D370 and ending at $D3E6. The address of these 11 items must be setup in a jump table found at locations $D3E7 thru $D3FB. A copy of the Custom I/O Driver Package used to produce FLEX.SYS is included at the end of the General Notes section. Use it as a guide for writing your own.

A description of each routine and vector follows.

INCH
The address of the input character routine should be placed at $D3FB. This routine should get one input character from the terminal and return it in 'A' with the parity bit cleared. It should also echo the character to the output device. Only 'A' and the condition codes may be modified.

OUTCH
The address of the output character should be placed at $D3F9. This routine should output the character found in 'A' to the output device. No registers should be modified except condition codes.

STAT
The address of the STAT routine should be placed at $D3F7. This routine checks the status of the input device. That is to say, it checks to see if a character has been typed on the keyboard. If so, a Not-Equal condition should be returned. If no character has been typed, an Equal to zero condition should be returned. No registers may be modified except condition codes.

TINIT
The address of the terminal initialization routine should be placed at $D3F5. This routine performs any necessary initialization for terminal I/O to take place. Any register may be modified except 'S'.

MONITR

This is the address to which execution will transfer when FLEX is exited. It is generally the reentry point of the system's monitor ROM. The address should be placed at $D3F3.

TMINT
The address of the timer initialization routine should be placed at $D3F1. This routine performs any necessary initialization for the interrupt timer used by the printer spooling process. Any register may be modified except 'S'.

TMON
The address of the timer on routine should be placed at $D3EF. This routines "turns the timer on" or in other words starts the interval IRQ interrupts. Any registers execpt 'S' may be modified.

TMOFF
The address of the timer off routine should be placed at $D3ED. This routine "turns the timer off" or in other words stops the interval IRQ interrupts. Any registers except 'S' may be modified.

IRQVEC
The IRQ vector is an address of a two byte location in RAM where FLEX can stuff the address of its IRQ interrupt handler routine. In other words, when an IRQ interrupt occurs control should be transferred to the address stored at the location specified by the IRQ vector. This IRQ vector location (address) should be placed at $D3EB.

SWIVEC
The SWI3 vector is an address of a two byte location in RAM where FLEX can stuff the address of its SWI3 interrupt handler routine. In other words, when an SWI3 interrupt occurs control should be transferred to the address stored at the location specified by the SWI3 vector. This SWI3 vector location (address) should be placed at $D3E9.

IHNDLR
The Interrupt Handler routine is the one which will be executed when an IRQ interrupt occurs. If using printer spooling, the routine should first clear the interrupt condtion and then jump to the 'change process' routine of the printer spooler at $C700. If not using printer spooling, this routine can be setup to do whatever the user desires. If it is desirable to do both printer spooling and have IRQ's from another device (besides the spooler clock), this routine would have to determine which device had caused the interrupt and handle it accordingly. The address of this routine should be placed at $D3E7.

The CUSTOM DISK DRIVER PACKAGE

This package supplies all the disk functions required by FLEX. There are eight routines in all:

READ	Reads a single sector
WRITE	Writes a single sector
VERIFY	Verifys a single sector
RESTORE	Restores the head to track 0
DRIVE	Selects the desired drive
CHECK	Checks a drive for a ready condition
QUICK	Same as CHECK but with no delay
INIT	Initializes any necessary values

These routines and what is required of them are decribed in the Advanced Programmer's Guide in the section titled 'DISK DRIVERS'. There is a jump table which contains the address of all these routines at $DE00. This table is as follows:

```
DE00      JMP  READ
DE03      JMP  WRITE
DE06      JMP  VERIFY
DE09      JMP  RESTOR
DEOC      JMP  DRIVE
DEOF      JMP  CHECK
DE12      JMP  QUICK
DE15      JMP  INIT
```

Immediately following this jump table there is a space for the disk driver routines. In the general case this space would start at $DE18 and run through $DFFF. In the SWTPc system with S-BUG installed, that entire space is not available due to the fact that S-BUG uses RAM in the area of $DFA0 to $DFFF for variables and stack. Thus the driver routine area is limited in this case to $DE18 through $DF9F.

The actual source listings for the SWTPc drivers are not included, but a skeletal Custom Disk Driver Package is included at the end of this section which should assist you in writing your own package.

PUTTING THE CUSTOM FLEX TOGETHER

Once you have written and assembled a Custom I/O and Custom Disk Driver packages, you are ready to append them to the core of FLEX (FLEX.COR) to produce a new, bootable version. This is done with the APPEND utility if FLEX, but before we get into that there is a very important point which must be covered.

*** IMPORTANT ***

The copy of FLEX on disk is much like any other standard binary file. IT MUST HAVE A TRANSFER ADDRESS IN ORDER TO WORK! It is also important to note that unlike other binary files FLEX can have ONLY ONE transfer address and it MUST BE THE LAST THING IN THE FILE! The simplest way of getting that transfer address into the file is by use of the END statement in the assembler. We recommend you put a transfer address on the END statement of the Custom I/O Driver Package and make sure it is the last thing in the final FLEX file.

Assuming you have put a transfer address on the Custom I/O Driver Package with an end statement of the form:

```
          END  $CD00
```

you can now create a new version of FLEX by appending the custom disk drivers and custom I/O drivers onto FLEX.COR. You should use the APPEND command for this purpose as shown:

```
     +++APPEND FLEX.COR DRVRS.BIN CUSTOMIO.BIN NEWFLEX.SYS
```

This command assumes the object file you created for the Custom Disk Drivers is called DRVRS.BIN and the Custom I/O Drivers are in a file called CUSTOMIO.BIN. The new, custom version of FLEX is called NEWFLEX.SYS. In order to boot up this NEWFLEX.SYS you must link it with the LINK command (see the FLEX User's and Advanced Progammer's Manuals). The command would be of the form:

```
     +++LINK NEWFLEX.SYS
```

The disk containing your newly made and linked FLEX can now be booted with the normal boot procedure.

```
        TTL FLEXADR
        STTL FLEX ADRESSEN
        OPT PAG
        PAG
* FLEX ADRESSEN

COLD    EQU     $CD00           COLDSTART 8
WARMS   EQU     $CD03           WARMSTART 8
RENTER  EQU     $CD06           RE-ENTRY POINT 8
INCH    EQU     $CD09           INPUT CHARACTER
INCH2   EQU     $CD0C           INPUT CHARACTER 9
OUTCH   EQU     $CD0F           OUTPUT CHARACTER 9
OUTCH2  EQU     $CD12           OUTPUT CHARACTER 9
GETCHR  EQU     $CD15           GET CHARACTER 9
PUTCHR  EQU     $CD18           PUT CHARACTER 10
INBUFF  EQU     $CD1B           INPUT INTO LINE-BUFFER 10
PSTRNG  EQU     $CD1E           PRINT STRING 10
CLASS   EQU     $CD21           CLASSIFY CHARACTER 11
PCRLF   EQU     $CD24           PRINT CRLF
NXTCH   EQU     $CD27           GET NEXT BUFFER CHARACTER 11
RSTRIO  EQU     $CD2A           RESTORE I/O VECTORS
GETFIL  EQU     $CD2D           GET FILE SPECIFICATION 12
LOAD    EQU     $CD30           FILE LOADER 12
SETEXT  EQU     $CD33           SET EXTENSION 13
ADDBX   EQU     $CD36           ADD B-REG TO X-REG 13
OUTDEC  EQU     $CD39           OUTPUT DECIMAL NUMBER 13
OUTHEX  EQU     $CD3C           OUTPUT HEX NUMBER 13
RPTERR  EQU     $CD3F           REPORT ERROR 14
GETHEX  EQU     $CD42           GET HEX NUMBER 14
OUTADR  EQU     $CD45           OUTPUT HEX ADDRESS
INDEC   EQU     $CD48           INPUT DECIMAL NUMBER 15
DOCMND  EQU     $CD4B           CALL DOS AS SUBROUTINE 15
STAT    EQU     $CD4E           CHECK TERMINAL INPUT STATUS 15

ULH     EQU     $DE1E           UNLOAD HEADS
MTROFF  EQU     $DE21           TURN OFF DRIVE MOTORS
FCTE    EQU     $DE24           FIND CONFIGURATION TABLE ENTRY
DEBCOU  EQU     $D3D1           DEBOUNCE COUNT
BLPER   EQU     $D3D3           BLINK PERIOD
MOBL    EQU     $D3D5           MOTOR OFF BLINK LIMIT
CURTYP  EQU     $D3D7           CURSOR TYPE
VDGMOD  EQU     $D3D9           VDG MODE
BELCYC  EQU     $D3E1           BELL TONE CYCLE COUNT
BELHLF  EQU     $D3E3           BELL TONE HALF PERIOD
INCHNE   EQU      $D4E5            INPUT    WITHOUT    ECHO    (JSR
[INCHNE])

TINIT   EQU     $D3F5           TERMINAL INITIALISATION 4
MONITR  EQU     $D3F1           MONITOR REENTRY 5
TMINT   EQU     $D3F1           TIMER INITIALISATION
TMON    EQU     $D3ED           TIMER ON
TMOFF   EQU     $D3ED           TIMER OFF
IRQVEC  EQU     $D3EB           IRQ-VECTOR

SWIVEC  EQU     $D3E9           SWI3-VECTOR
```

```
IHNDLR  EQU     $D3E7           IR-HANDLER

* DOS MEMORY MAP

LINBUF  EQU     $C080           128 BYTE LINE BUFFER
TTYBS   EQU     $CC00           TTY BACKSPACE CHR ($08)
TTYDEL  EQU     $CC01           TTY DELETE CHR ($18)
TTYEOL  EQU     $CC02           ":" ($3A)
TTYDEP  EQU     $CC03           ZEILENANZAHL (0)
TTYWID  EQU     $CC04           SPALTENANZAHL (0)
TTYNUL  EQU     $CC05           NULL-COUNT (4)
TTYTAB  EQU     $CC06           TAB-CHARACTER (0)
TTYECH  EQU     $CC07           BS-ECHO CHARACTER (0)
TTYEJC  EQU     $CC08           BLANK-LINES (0)
TTYPAC  EQU     $CC09           PAUSE CONTROL ($FF)
TTYESC  EQU     $CC0A           ESCAPE CHARACTER ($1B)
SYSDRI  EQU     $CC0B           SYSTEM-DRIVE NUMBER
WORDRI  EQU     $CC0C           WORKING-DRIVE NUMBER
SYSSCR  EQU     $CC0D           SYSTEM SCRATCH
MONTH   EQU     $CC0E           SYSTEM
DAY     EQU     $CC0F           DATE
YEAR    EQU     $CC10           REGISTERS
LASTERM EQU     $CC11           LAST TERMIATOR
USCOMTA EQU     $CC12           USER COMMAND TABLE ADRES
LIBUPO  EQU     $CC14           LINE BUFFER POINTER
ESCRET  EQU     $CC16           ESCAPE RETURN REGITER
CURCHR  EQU     $CC18           CURRENT CHARACTER
PRECHR  EQU     $CC19           PREVIOUS CHARACTER
CURLIN  EQU     $CC1A           CURRENT LINE NUMBER
LOADOFF EQU     $CC1B           LOADER ADDRESS OFFSET
TRFFLAG EQU     $CC1D           TRANSFER FLAG
TRFADDR EQU     $CC1E           TRANSFER ADDRESS
ERRTYP  EQU     $CC20           ERROR TYPE
SPIOFLA EQU     $CC21           SPECIAL I/O FLAG
OUTSWI  EQU     $CC22           OUTPUT SWITCH
INPSWI  EQU     $CC23           INPUT SWITCH
FOUTADD EQU     $CC24           FILE OUTPUT ADDRESS
FINPADD EQU     $CC26           FILE INPUT ADDRESS
CMNDFL  EQU     $CC28           COMMAND FLAG
CURCOL  EQU     $CC29           CURRENT OUTPUT COLOMN
MEMEND  EQU     $CC2B           MEMORY END
ERRVEND EQU     $CC2D           ERROR NAME VENDOR
FECHO   EQU     $CC2F           FILE INPUT ECHO FLAG
SYSCONS EQU     $CC4E           SYSTEM CONSTANTS
PRINI   EQU     $CCC0           PRINTER INITIALIZE
PRREADY EQU     $CCD8           PRINTER READY CHECK
PROUT   EQU     $CCE4           PRINTER OUTPUT

* FILE MANAGEMENT SYSTEM

FMSINIT EQU     $D400           FMS INITIALIZATION
FMSCLOS EQU     $D403           FMS CLOSE
FMSCALL EQU     $D406           FMS CALL
FCBBASE EQU     $D40A           FCB BASE POINTER
```

```
FCBCURR EQU     $D40B           CURRENT FCB ADDRESS
VERFLAG EQU     $D435           VERIFY FLAG

* CSSADR.TXT* flex 9 addresses and constants
*
flxorg equ $c000
*
sstack equ flxorg+$007f flex stack
lnbuff equ flxorg+$0080 line buffer
startr equ flxorg+$0100 utility area
*
sysfcb equ flxorg+$0840 fcb function code
sferor equ flxorg+$0841 fcb error status
sfactv equ flxorg+$0842 fcb activity status
sfdriv equ flxorg+$0843 fcb drive number
sfname equ flxorg+$0844 fcb name
sfextn equ flxorg+$084c fcb extension
sfattr equ flxorg+$084f fcb file attributes
sfres1 equ flxorg+$0850 fcb reserved
sfsadr equ flxorg+$0851 fcb starting disk address
sfeadr equ flxorg+$0853 fcb ending disk address
sfsize equ flxorg+$0855 fcb file size
sffscm equ flxorg+$0857 fcb file sector map indicator
sfres2 equ flxorg+$0858 fcb reserved
sfdmon equ flxorg+$0859 fcb creation month
sfdday equ flxorg+$085a fcb creation day
sfdyer equ flxorg+$085b fcb creation year
sflist equ flxorg+$085c fcb list pointer
sfcrpo equ flxorg+$085e fcb current position
sfcrec equ flxorg+$0860 fcb current record number
sfindx equ flxorg+$0862 fcb data index
sfrinx equ flxorg+$0863 fcb random index
sfnwbf equ flxorg+$0864 fcb name work buffer
sfcdir equ flxorg+$086f fcb current directory address
sfdelp equ flxorg+$0872 fcb first deleted directory pointer
sfrenm equ flxorg+$0875 fcb rename work area
sfcomp equ flxorg+$087b fcb space compression flag
sfbuff equ flxorg+$0880 fcb sector buffer
*
ttybsp equ flxorg+$0c00 ttyset backspace
ttydel equ flxorg+$0c01 ttyset delete
ttyeol equ flxorg+$0c02 ttyset end of line
ttydep equ flxorg+$0c03 ttyset depth count
ttywid equ flxorg+$0c04 ttyset width count
ttynlc equ flxorg+$0c05 ttyset null count
ttytab equ flxorg+$0c06 ttyset tab
ttybec equ flxorg+$0c07 ttyset backspace echo
ttyejc equ flxorg+$0c08 ttyset eject count
ttypau equ flxorg+$0c09 ttyset pause control
ttyesc equ flxorg+$0c0a ttyset escape
sysdrv equ flxorg+$0c0b system drive number
wrkdrv equ flxorg+$0c0c working drive number
sysflg equ flxorg+$0c0d use system drive flag
smonth equ flxorg+$0c0e system month
sday   equ flxorg+$0c0f system day
```

```
syearn equ flxorg+$0c10 system year
lsttrm equ flxorg+$0c11 last terminator
  coldst equ flxorg+$0d00 cold start
  warmst equ flxorg+$0d03 warm start
  renter equ flxorg+$0d06 re-entry
  inchar equ flxorg+$0d09 basic input character
  incha2 equ flxorg+$0d0c basic input character
  outchr equ flxorg+$0d0f basic output character
uctaba equ flxorg+$0c12 user command table
lnbufp equ flxorg+$0c14 line buffer pointer
escrtn equ flxorg+$0c16 escape return
curchr equ flxorg+$0c18 current character
prvchr equ flxorg+$0c19 previous character
curlin equ flxorg+$0c1a current line number
loadof equ flxorg+$0c1b loader address offset
trflag equ flxorg+$0c1d transfer flag
traddr equ flxorg+$0c1e transfer address
errtyp equ flxorg+$0c20 error type
spciof equ flxorg+$0c21 special i/o flag
outswt equ flxorg+$0c22 output switch
inswtc equ flxorg+$0c23 input switch
foaddr equ flxorg+$0c24 file output address
fiaddr equ flxorg+$0c26 file input address
comflg equ flxorg+$0c28 command flag
crotcl equ flxorg+$0c29 current output column
syssc2 equ flxorg+$0c2a system scratch
memend equ flxorg+$0c2b memory end
errvec equ flxorg+$0c2d error name vector
fieflg equ flxorg+$0c2f file input echo flag
syssc3 equ flxorg+$0c30 system scratch
syssc4 equ flxorg+$0c31 system scratch
syssc5 equ flxorg+$0c32 system scratch
cputyp equ flxorg+$0c33 cpu type flag
cp2mhz equ $80 2mh clock
cpslow equ $40 memory stretch
cp50hz equ $20 50 hz power
cpramf equ $10 cpu ram available
cprtck equ $08 6819 rtc available
cpiobx equ $04 4 addr/port
cptime equ $02 6840 timer available
cpxmem equ $01 extended memory used
ptrrap equ flxorg+$0c35 reserved printer area pointer
ptrral equ flxorg+$0c37 reserved printer area length
ptrdev equ flxorg+$0c39 printer device address
retadr equ flxorg+$0c43 docmd return address
ulcflg equ flxorg+$0c49 upper/lower case flag
prompt equ flxorg+$0c4e pointer to prompt string
prinit equ flxorg+$0cc0 printer initialization
prterm equ flxorg+$0cd0 printer close routine
prdych equ flxorg+$0cd8 printer ready check
ptrout equ flxorg+$0ce4 printer output
prcflg equ flxorg+$0cfc active spooling flag
syssc6 equ flxorg+$0cf8 system scratch
```

```
outch2 equ flxorg+$0d12 basic output character
getchr equ   xorg+$0d15 get character


putchr equ flxorg+$0d18 put character
inbuff equ flxorg+$0d1b input into line buffer
pstrng equ flxorg+$0d1e print string
clasfy equ flxorg+$0d21 classify character
prcrlf equ flxorg+$0d24 print crlf
nxtchr equ flxorg+$0d27 get next buffer character
rstrio equ flxorg+$0d2a restore i/o vectors
getfil equ flxorg+$0d2d get file specs
loadfl equ flxorg+$0d30 load binary file
setext equ flxorg+$0d33 set extension
addb2x equ flxorg+$0d36 add b to x
outdec equ flxorg+$0d39 output decimal number
outhex equ flxorg+$0d3c output hexadecimal number
rpterr equ flxorg+$0d3f report error
gethex equ flxorg+$0d42 get hexadecimal number
outadr equ flxorg+$0d45 output hexadecimal address
indecm equ flxorg+$0d48 input decimal number
docmnd equ flxorg+$0d4b call dos as a suboutine
ctstat equ flxorg+$0d4e check terminal status
*
tapptr equ flxorg+$13de vector for input tap
tapdum equ flxorg+$13e0 dummy rts for rm
setirq equ flxorg+$13e1 clear irq vector
clrirq equ flxorg+$13e3 set irq vector
trminp equ flxorg+$13e5 terminal input without echo
timoff equ flxorg+$13ed timer off
timron equ flxorg+$13ef timer on
timint equ flxorg+$13f1 timer init
trmint equ flxorg+$13f5 terminal init
trmchk equ flxorg+$13f7 terminal check
trmout equ flxorg+$13f9 terminal output
trmine equ flxorg+$13fb terminal input with echo
*
fmsint equ flxorg+$1400 fms initialization
fmscls equ flxorg+$1403 fms close files
fmscal equ flxorg+$1406 fms call
fcbbas equ flxorg+$1409 fcb base pointer
fcbcur equ flxorg+$140b fcb current address
verflg equ flxorg+$1435 verify flag
*
dreadr equ flxorg+$1e00 basic read disk
dwrite equ flxorg+$1e03 basic write disk
dverfy equ flxorg+$1e06 basic verify disk
drstor equ flxorg+$1e09 basic restore disk
ddselr equ flxorg+$1e0c basic select drive
dcheck equ flxorg+$1e0f basic check drive ready
dqchek equ flxorg+$1e12 basic quick drive check
dseekt equ flxorg+$1e1b basic drive seek-to-sector
*
fmgnxb equ $00 get next byte
fmpnxb equ $00 put next byte
fmopni equ $01 open for input
fmopno equ $02 open for output
fmopnu equ $03 open for update
```

```
fmclsi equ $04 close for input
fmclso equ $04 close for output
fmclsu equ $04 close for update
fmrewf equ $05 rewind file
fmopdi equ $06 open directory
fmgeti equ $07 get information record
fmputi equ $08 put information record
fmrdss equ $09 read single sector
fmwtss equ $0a write single sector
fmresd equ $0b reserved
fmdelt equ $0c delete file
fmrenm equ $0d rename file
fmrese equ $0e reserved
fmnxsq equ $0f get next sequential sector
fmopir equ $10 open system information record
fmgtrb equ $11 get random byte from sector
fmptrb equ $12 put random byte into sector
fmresf equ $13 reserved
fmnxdr equ $14 find next drive
fmrecn equ $15 position to record n
fmback equ $16 back up one record
*
fcfunc equ $00 fcb function code
fceror equ $01 fcb error status
fcactv equ $02 fcb activity status
fcdriv equ $03 fcb drive number
fcname equ $04 fcb name
fcextn equ $0c fcb extension
fcattr equ $0f fcb file attributes
fcres1 equ $10 fcb reserved
fcsadr equ $11 fcb starting disk address
fceadr equ $13 fcb ending disk address
fcsize equ $15 fcb file size
fcfscm equ $17 fcb file sector map indicator
fcres2 equ $18 fcb reserved
fcdmon equ $19 fcb creation month
fcdday equ $1a fcb creation day
fcdyer equ $1b fcb creation year
fclist equ $1c fcb list pointer
fccrpo equ $1e fcb current position
fccrec equ $20 fcb current record number
fcindx equ $22 fcb data index
fcrinx equ $23 fcb random index
fcnwbf equ $24 fcb name work buffer
fccdir equ $2f fcb current directory address
fcdelp equ $32 fcb first deleted directory pointer
fcrenm equ $35 fcb rename work area
fccomp equ $3b fcb space compression flag
fcbuff equ $40 fcb sector buffer
*
frillg equ 01 illegal fms function code encountered
frinus equ 02 the requested file is in use
frexis equ 03 the file specified already exists
frabsn equ 04 the specified file could not be found
frsysd equ 05 system directory error - reboot system
frfuld equ 06 the system directory space is full
frnosp equ 07 all available disk space has been used
frendf equ 08 read past end of file
frrder equ 09 disk file read error
frwter equ 10 disk file write error
frwrpr equ 11 the file or disk is write protected!
frprot equ 12 the file is protected - file not deleted
frbfcb equ 13 illegal file control block specified
frbadr equ 14 illegal disk address encountered
frbdrv equ 15 an illegal drive number was specified
frdrnr equ 16 drives not ready
frflpr equ 17 the file is protected - access denied
frsyss equ 18 system file status error
frdtix equ 19 fms data index range error
frfmse equ 20 fms inactive - reboot system
frilfl equ 21 illegal file specification
frsysc equ 22 system file close error
frsmap equ 23 sector map overflow - disk too segmented
frbrno equ 24 nonexistent record number specified
frbfil equ 25 record number match error - file damaged
frsntx equ 26 command syntax error - retype command
frprnt equ 27 that command is not allowed while printing
frhard equ 28 wrong hardware configuration
frundf equ 29 undefined
*
extbin equ $00 .bin
exttxt equ $01 .txt
extcmd equ $02 .cmd
extbas equ $03 .bas
extsys equ $04 .sys
extbak equ $05 .bak
extscr equ $06 .scr
extdat equ $07 .dat
extbac equ $08 .bac
extdir equ $09 .dir
extprt equ $0a .prt
extout equ $0b .out
*
*
```