6809 FLEX™ For UniFLEX™

technical systems consultants, inc.

6809 FLEX™ For UniFLEX™

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, North Carolina 27514

MANUAL REVISION HISTORY

Revision	Date	Change
A	3/84	Original Release

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

IMPORTANT NOTICE

The current release of "FLEX for UniFLEX" supports neither 5 1/4-inch floppy disks nor 8-inch double-density floppy disks.

1. INTRODUCTION

This manual describes the use and operation of a version of the FLEX Disk Operating System which runs as a task under the UniFLEX Operating System. "FLEX for UniFLEX" allows almost full use of the FLEX Disk Operating System, including the reading and writing of floppy disks which have been formatted for use under the FLEX Operating System (FLEX-formatted floppy disks). The manual assumes that the reader is familiar with the operation of both UniFLEX and FLEX.

Since the FLEX Disk Operating System is really only a program that runs on the 6809 microprocessor, it is possible to run it on UniFLEX with very minor changes. These changes are all associated with FLEX's communication with the terminal and disks. Programs which run under UniFLEX cannot directly access hardware devices, such as terminals and printers. All hardware communication must be accomplished through UniFLEX system calls. "FLEX for UniFLEX", therefore, has been modified so that it uses UniFLEX system calls to communicate with the terminal and disks. Because this modification is transparent to the user, it appears that the user is running a standard version of FLEX.

There is one caveat. Since the hardware is not directly accessible to a UniFLEX program (remember "FLEX for UniFLEX" is a normal UniFLEX program), FLEX-based programs which directly access a hardware device, such as an ACIA or PIA, cannot be used. All I/O must be routed through standard FLEX routines such as PSTRNG and PCRLF. Thus, programs such as printer drivers or PROM programming routines will probably not function correctly.

The first-time user should read this manual entirely, then return to Section 2.4 for detailed instructions on the startup procedure.

This manual uses several notational conventions in the discussions of command syntax. Items that are not enclosed in angle brackets, '<' or '>', or square brackets, '[' and ']', must be typed as shown. Angle brackets enclose descriptions which the user must replace with a specific argument. Expressions enclosed only in angle brackets are essential parts of the command line. Square brackets indicate optional items. These items may be omitted if their effect is not desired. The underscore character, '_' is used to link separate words which describe one term, such as "UniFLEX" and "command".

The "FLEX for UniFLEX" package is supplied with the standard FLEX user's manual. The "FLEX Advanced Programmer's Guide" is not presently included. The entire set of FLEX manuals can be purchased separately from Technical Systems Consultants if desired. Note also that "FLEX for UniFLEX" does not include an editor or assembler.

2. RUNNING "FLEX FOR UNIFLEX"

FLEX can support up to four disks, numbered 0 through 3. When "FLEX for UniFLEX" is run, various UniFLEX floppy disk drives must be assigned to operate as FLEX disk drives until FLEX is terminated. While they are assigned as FLEX drives, no user should mount them as UniFLEX drives. A special feature permits "FLEX for UniFLEX" to treat a specially created UniFLEX file like a FLEX disk. This manual refers to such a file as a file-disk. These file-disks consist of 256-byte records of data, much like the 256-byte sectors of a real disk. Such a UniFLEX file would normally be created on a hard disk, but it is possible to create one on a UniFLEX floppy disk if space permits. Instructions for creating file-disks are given in the documentation for the FNEWDISK command, found later in this manual.

2.1 Calling FLEX

The simplest UniFLEX command necessary to invoke ("boot up") "FLEX for UniFLEX" is as follows:

```
++ FLEX +0=<drive_spec>
```

The two plus signs are UniFLEX's ready prompt, and FLEX is the name of the UniFLEX-resident version of FLEX. The purpose of the '0' (zero) option is to specify which physical UniFLEX drive or file-disk to use as logical FLEX drive 0. The parameter <drive_spec> can be one of three things:

1. A single digit indicating the UniFLEX drive number.
2. The name of the UniFLEX disk device, not including the path specification. For example, "fd1" specifies floppy drive 1.
3. The name of a UniFLEX file which has been established as a file-disk.

It is possible to assign more than a single drive to FLEX by using a command similar to the following one:

```
++ FLEX +0=<drive_spec_0> +1=<drive_spec_1>  +2=<drive_spec_2>
```

Note that a separate option defines each FLEX drive. The plus sign, '+', is immediately followed by the FLEX drive number, which must be an integer from 0 to 3 inclusive. The FLEX drive number is followed by an optional equals sign, '=', and a <drive_spec> (described earlier). Any combination of the four possible FLEX drives can be assigned to any combination of UniFLEX drives or file-disks, as long as FLEX drive 0 is always defined.

"FLEX for UniFLEX" does not prompt the user for the current date when it is loaded. Instead, it automatically obtains the current date from UniFLEX. The STARTUP feature of FLEX functions normally. Thus, if FLEX drive 0 is defined as a floppy disk drive, the system disk should be inserted in that drive before FLEX is called.

2.2 UniFLEX Systems with only Floppy Disks

On a UniFLEX system equipped only with floppy disks, the UniFLEX system disk must always be in "fd0", even while "FLEX for UniFLEX" is running. If only two drives are available, the FLEX system disk must go in "fd1". Such a system is, in effect, a single-drive FLEX system.

Several examples of running "FLEX for UniFLEX" follow. They all assign UniFLEX drive "fd1" to FLEX drive 0.

```
++ FLEX +0=1
++ FLEX +0=fd1
++ FLEX +01
++ FLEX +0fd1
```

2.3 UniFLEX Systems with a Hard Disk

On a UniFLEX system with a hard disk as the root device, it is possible to place FLEX-formatted disks into both "fd0" and "fd1". The following four equivalent commands all map FLEX drive 0 onto UniFLEX "fd0" and FLEX drive 1 onto UniFLEX "fd1":

```
++ FLEX +0=0 +1=1
++ FLEX +0=fd0 +1=fd1
++ FLEX +00 +11
++ FLEX +0fd0 +1fd1
```

Conversely, it is possible to map FLEX drive 0 onto UniFLEX "fd1" and FLEX drive 1 onto UniFLEX drive "fd0" by using any of the following commands:

```
++ FLEX +0=1 +1=0
++ FLEX +0=fd1 +1=fd0
++ FLEX +01 +10
++ FLEX +0fd1 +0fd2
```

If a file-disk called "/flexdisk" has been created on the UniFLEX hard disk, the user can run FLEX using this file-disk as FLEX drive 0 and UniFLEX drive "fd1" as FLEX drive 1 by issuing any of the following commands:

```
++ FLEX +0=/flexdisk +1=fd1
++ FLEX +0=/flexdisk +1=1
++ FLEX +0=/flexdisk +11
++ FLEX +0/flexdisk +11
```

2.4 The Start-up Procedure

The software for "FLEX for UniFLEX" is supplied on a floppy disk formatted for use under UniFLEX (a UniFLEX-formatted floppy disk). This disk contains two important files: the FLEX command (which is the FLEX program) and a file-disk named "flexdisk". The file-disk looks like a normal FLEX disk to "FLEX for UniFLEX" even though it really is a UniFLEX file. It contains all the standard FLEX commands as well as some additional commands that are unique to "FLEX for UniFLEX". The file-disk must be copied to the root directory on the UniFLEX system disk so that it can subsequently be accessed by "FLEX for UniFLEX".

The first step in using "FLEX for UniFLEX" is to copy it onto the UniFLEX system disk. This is done with the normal UniFLEX insert procedure:

1. Boot UniFLEX.
2. Insert the master "FLEX for UniFLEX" floppy disk into "fd1".
3. Mount the disk with the command "/etc/mount /dev/fd1 /usr2".
4. Enter the command "/usr2/insert" and wait for its completion.
5. Unmount the floppy with the command "/etc/unmount /dev/fd1".
6. Remove the master "FLEX for UniFLEX" floppy disk and store it in a safe place.

After completing this procedure, the user can run "FLEX for UniFLEX" by entering one of the command lines shown in Sections 2.1 through 2.3. Without any other FLEX disk, "FLEX for UniFLEX" can be run with the command:

```
++ FLEX +0=/flexdisk
```

To save space on the system disk, the user may wish to copy all the FLEX files in the file-disk "/flexdisk" to a FLEX-formatted floppy disk. The file "/flexdisk" can then be deleted. The following procedure accomplishes this:

1. Boot FLEX with the command "FLEX +0=/flexdisk +1=1"
2. If a preformatted FLEX disk is not available, format a blank disk by inserting it in "fd1" and typing the FLEX command "NEWDISK 1".
3. Copy all FLEX files from the file-disk "flexdisk" to the freshly formatted floppy disk with the command "COPY 0 1".
4. Return to UniFLEX with the MON command and delete the file-disk with the command "kill /flexdisk".

3. OPERATING IN THE UNIFLEX ENVIRONMENT

Because "FLEX for UniFLEX" runs as a task under UniFLEX and calls upon UniFLEX to perform any I/O, the configuration of the UniFLEX terminal drivers is important. When "FLEX for UniFLEX" is run, it automatically configures the terminal drivers as necessary; it automatically reconfigures them for UniFLEX when it terminates. Some slight conflicts do exist, however, between FLEX and UniFLEX terminal I/O. They are discussed in this section.

3.1 Terminal Configuration under "FLEX for UniFLEX"

Standard FLEX appends a line-feed character to each carriage return it prints. In the normal terminal configuration, UniFLEX also appends line-feed characters to carriage returns. Since UniFLEX system calls are used to perform FLEX I/O, a normally configured UniFLEX terminal would receive two line-feeds for each carriage return printed by a FLEX program. The result would be double spacing. To prevent double spacing, "FLEX for UniFLEX" uses a "ttyset" call with the parameter "-alf" to configure the UniFLEX terminal drivers so that they do not append line-feeds to carriage returns. When the user exits FLEX, the terminal drivers are restored to their original configuration.

A side effect of this approach is that when FLEX sets the UniFLEX terminal configuration to "-alf" (turns off line-feeds), any messages sent to the terminal by UniFLEX do not have the necessary line-feed characters appended to the carriage returns. Consequently, the lines of the message are all printed on the same line, one on top of the other. Overprinting occurs, for example, when another user "sends" a message to a FLEX user, or when a UniFLEX assembly running in the background terminates (see Section 5). This approach is necessary to maintain compatibility with existing FLEX programs.

3.2 Abnormal Termination of FLEX

Under normal circumstances, FLEX is terminated with the MON command. Such a termination properly resets the UniFLEX terminal configuration to the configuration that existed when FLEX was called. FLEX may, however, terminate abnormally, as in the case of a kill interrupt sent by the system manager. If FLEX terminates abnormally, it bypasses the routines which reset the UniFLEX terminal configuration and leaves the UniFLEX terminal in an improper state. Two things are usually wrong: the terminal does not echo characters, nor does it generate a line-feed character after each carriage return. A terminal in this state appears rather lifeless.

If FLEX is abnormally terminated and the terminal appears to be incorrectly configured, a single UniFLEX command usually corrects the problem:

```
++ ttyset +echo +alf
```

This command the turns on both echoing and automatic generation of line-feed characters. Note that the characters are not echoed to the terminal as the user types the command.

3.3 Use of the Escape Key, FLEX Escape Character, and End-of-Page Pause

Under UniFLEX, typing the escape key temporarily halts output to the terminal. Typing the escape key again resumes the output. Since "FLEX for UniFLEX" is actually a UniFLEX program, typing the escape key also halts output being generated by FLEX, regardless of how the FLEX pause character is defined. The UniFLEX escape key is defined as an ASCII ESC (hexadecimal 1B). Currently, the user cannot redefine it.

FLEX also has an escape mechanism for temporarily halting output, but the user can redefine the escape character with the FLEX TTYSET command. character. As in UniFLEX, the default is ASCII ESC (hexadecimal 1B). In addition to stopping and restarting output, the FLEX escape character resumes output after an "end-of-page pause" has automatically suspended output at the end of a page. The "end-of-page pause" is enabled by specifying the appropriate PS and DP paramters to the FLEX TTYSET command.

A conflict arises, however, when the FLEX escape character is an ASCII ESC (the default) and the FLEX "end-of-page pause" is enabled. The UniFLEX terminal drivers trap all escape characters (ASCII ESC) and never pass them to the running program. Therefore, FLEX can never receive an ASCII ESC from the keyboard. If the FLEX escape character is an ASCII ESC and an "end-of-page pause" occurs, it is impossible to resume the output. The only recourse is to type a carriage return, which the FLEX "end-of-page pause" feature accepts as a signal to jump to the location stored in the FLEX Return Register, usually terminating the program.

It should be clear that the FLEX TTYSET escape feature does not function if the escape character is an ASCII ESC. Typing the escape key (ASCII ESC) does stop output, but UniFLEX, not FLEX, performs that stop. In most cases this is satisfactory, if not preferable. If the FLEX "end-of-page pause" feature is not required, the recommended action is to leave it disabled (which is the default) and to leave the FLEX escape character defined as ASCII ESC (also the default). However, if the user wants to use the FLEX "end-of-page pause", the FLEX escape character must be redefined to some value other than ASCII ESC (the default).

For example, suppose the FLEX "end-of-page pause" feature is required. The feature must be enabled with a FLEX command similar to the following one:

```
+++TTYSET PS=Y DP=23
```

In addition, the user must redefine the escape character to something other than ASCII ESC. The following command defines it as control-F (hexadecimal 06):

```
+++TTYSET ES=06
```

The UniFLEX escape character is usually typed to temporarily halt the listing of a long file to the terminal. Another UniFLEX escape character is typed to resume the listing. The user can also pause and resume output by typing the FLEX escape character, now defined as a control-F. When an "end-of-page pause" occurs, however, the UniFLEX escape character does not resume output. In such a case the user must type the FLEX escape character, which in this case is control-F. It is still usually possible to terminate the output immediately when a FLEX "end-of-page pause" occurs by typing a carriage return.

4. SPECIAL CONSIDERATIONS

While "FLEX for UniFLEX" is, on the whole, a complete, standard implementation of FLEX, the user must keep a few considerations in mind.

4.1 Printing from FLEX

The user cannot send data directly to a printer under "FLEX for UniFLEX" because, under UniFLEX, the printer-driver routines which normally perform the printing from FLEX cannot directly access the printer hardware. Instead, all printing must be done through the facilities of UniFLEX. The recommended procedure is first to prepare a FLEX disk-file containing the data to be printed. This may be done in a number of ways. If the program producing output cannot directly create an output file, the most common method is to use the FLEX "O" command. The "O" command redirects the output, which normally goes to the terminal, to a disk-file. The "O" command can be used anywhere the "P" command would be used under other FLEX installations.

Once the data to be printed are in a FLEX file, the file can be copied to UniFLEX by using the COPYFU command (see Section 5). The resulting UniFLEX file can then be printed using a normal UniFLEX printer-spooler command. A printer spooler is especially useful because it avoids printer conflicts with other users of the system, be they "FLEX for UniFLEX" or UniFLEX users. The UniFLEX printer-spooler command can be executed without leaving FLEX by using the FLEX "X" command (see Section 5). Finally, it may be desirable to delete the data files.

This procedure can be summarized in the following example involving the FLEX program REPORT, which prints a report to the terminal. Under other FLEX implementations, the user could route this report to the printer with the FLEX command:

```
+++P REPORT
```

Under "FLEX for UniFLEX" the user cannot route directly to the printer with the "P" command, so the following procedure might be used:

```
+++O 1.TEMP.OUT REPORT
+++COPYFU 1.TEMP.OUT /tmp/flex_temp
+++X spr /tmp/flex_tmp
+++DELETE 1.TEMP.OUT
+++U kill /tmp/flex_tmp
```

This procedure shows what commands to type, but, for clarity, it does not show all the resulting prompts and messages. The first line executes REPORT and places the output in the FLEX file 1.TEMP.OUT. The second line copies this file to the UniFLEX disk system, calling it

"/tmp/flex_temp". The third line calls the UniFLEX printer spooler (in this example it is a serial printer spooler called "spr"), and requests it to print the file "/tmp/flex_tmp". The last two commands delete the now unnecessary temporary files. Note that the UniFLEX printer-spooler command is executed in the background (see Section 5), so it is not necessary to wait for the printing to complete before proceeding with other tasks.

An accomplished UniFLEX programmer might wish to develop a set of FLEX printer driver routines which send data directly to UniFLEX (but not directly to the hardware). It is certainly possible to do so as long as the normal rules for FLEX printer drivers are followed. The FLEX "P" command has been included with this package to accommodate such use. No PRINT.SYS file is supplied with this package.

4.2 General

The items in the following list do not belong in any single category. They are various suggestions, reminders, and warnings to users of "FLEX for UniFLEX".

1. FLEX disks cannot be "mounted" under UniFLEX.

2. FLEX disks can be removed from the drives anytime as long as the head is not loaded and the FLEX "+++" prompt is waiting.

3. FLEX itself is loaded from a UniFLEX disk, not from the FLEX system disk. However, the FLEX system disk should be inserted in the drive defined as drive 0 before executing the FLEX command from UniFLEX. This procedure allows the FLEX STARTUP procedure to function normally.

4. The MON command returns the user to UniFLEX.

5. FLEX programs which directly access I/O hardware may not be used.

6. Memory above $F000 is reserved for use by UniFLEX. User programs should not alter this area.

7. Typing a control-C during the execution of any FLEX utility or program causes a return to the FLEX escape/return register ($CC16), which is usually set to WARMS ($CD03).

8. FLEX programs must NOT set the IRQ interrupt mask. Doing so may cause an unprotected UniFLEX system to crash.

9. Double-density FLEX-formatted disks are not supported.

10. FLEX printer spooling cannot be used. The associated commands, PRINT and QCHECK are not included. Section 4.1 explains how to print from "FLEX for UniFLEX".

11. Disks formatted with the NEWDISK command from "FLEX for UniFLEX" cannot boot other FLEX systems.

5. ADDITIONAL FLEX UTILITIES

This package includes seven new FLEX utilities, unique to "FLEX for UniFLEX". These utilities permit access to files and commands on the UniFLEX system without leaving FLEX. Brief descriptions of these utilities follow.

DRIVES	Report which UniFLEX drives or file-disks are assigned to the four possible FLEX drives.
U	Execute a UniFLEX command without leaving FLEX.
X	Execute a single UniFLEX command in the background without leaving FLEX. FLEX continues to run concurrently with the UniFLEX task.
WAIT	Wait for a UniFLEX background task (started with the "X" command) to complete.
COPYFU	Copy a file from a FLEX disk to a UniFLEX disk.
COPYUF	Copy a file from a UniFLEX disk to a FLEX disk.
FNEWDISK	Create a UniFLEX file-disk.

These utilities are described in more detail in the following pages. The user may want to remove these pages and insert them into the section about utilities in the operating system manual for FLEX.

DRIVES

The DRIVES command provides a list of the four FLEX drive numbers and the UniFLEX drives or "file-disks" associated with each one. This command provides a useful reminder of what resources are available to the user of "FLEX for UniFLEX".

DESCRIPTION

The syntax of the DRIVES command is simply

```
DRIVES
```

There are no parameters or options.

U

The "U" command permits the execution of a UniFLEX command from FLEX. It eliminates the need to return to UniFLEX before executing a UniFLEX command. The specified command is passed to the UniFLEX shell program for execution. The "U" command waits for the UniFLEX command to finish before prompting for another FLEX command.

DESCRIPTION

The syntax of the "U" command is

```
U <UniFLEX_command>
```

where <UniFLEX_command> is a standard UniFLEX command. Everything on the command line beyond the "U", including spaces and punctuation, is passed directly to UniFLEX. Remember that UniFLEX generally expects commands and file names to be written in lowercase characters. Some examples of the use of the "U" command follow:

```
+++U dir +1
+++1.U asmb /usr/john/source +bls
+++U (who;date) >out-file
```

The first command produces a directory listing of the UniFLEX working directory (the one which the user was in when FLEX was called). The second example loads "U" from FLEX drive 1 and runs the UniFLEX assembler on the file "/usr/john/source". The third example performs the "who" and "date" commands, groups their output, and redirects it to the file "out-file".

X

The "X" command is similar to the "U" command except "X" is used to execute a UniFLEX task in the background. The UniFLEX command is passed to the shell program for execution. Unlike "U", "X" does not wait for the background task to complete; it immediately returns the FLEX prompt. A task initiated by "X" must be completed before another background task may be started by "X". Also, the WAIT command must be used to receive the termination status of a task before the "X" command may be used again.

DESCRIPTION

The syntax of the "X" command is

```
X <UniFLEX_command>
```

where <UniFLEX_command> is a standard UniFLEX command. Everything on the command line beyond the "X", including spaces and punctuation, is passed directly to UniFLEX. Remember that UniFLEX generally expects commands and file names to be written in lowercase characters. Some examples of the use of the "X" command follow:

```
+++X asmb /dat/act-receive +ls >/dat/asmerrs
+++X list test1 test2 >all-tests
+++X copy /flexdisk /usr2/backup
```

The first command starts a UniFLEX assembly in the background. Because it redirects standard output to the file "/dat/asmerrs", any error messages reported during the UniFLEX assembly are sent to this file rather than to the terminal. The user can, therefore, continue to operate "FLEX for UniFLEX" without being interrupted by error messages from the assembler. Later, the output can be examined to see if any errors occurred. The second example lists two files and redirects the output to a single file. The last example starts a background job that copies the file "/flexdisk" to the file "/usr2/backup".

WAIT

The WAIT command is used to wait for the completion of a background task generated by the "X" command. This command cannot be used to wait for completion of a background task that was not generated by the "X" command. The FLEX prompt does not appear until the background task is complete. When the background task terminates, a message is displayed specifying the task number and whether or not it terminated normally. In the event of abnormal termination, the message includes the response code or interrupt code that caused the termination.

DESCRIPTION

The syntax of the WAIT command is simply

```
WAIT
```

There are no parameters or options. If no background job exists, the user is so informed.

COPYFU

The COPYFU command copies a file from a FLEX disk to a UniFLEX disk.

DESCRIPTION

The syntax of the COPYFU command is

```
+++COPYFU <FLEX_file_name> <UniFLEX_file_name> [+R]
```

where the extension for the FLEX file name defaults to TXT. The UniFLEX file name can be any standard UniFLEX file specification. The 'R' option tells COPYFU to copy the file in raw mode--that is, character-by-character without any changes. If the user does not specify the 'R' option, the FLEX file is assumed to be a text file, and space expansion is performed on it as it is read. All binary files and most data files should be copied with the 'R' option. Some examples of COPYFU follow:

```
+++COPYFU CHAPTER1 /book/chapter1
+++COPYFU 2.INVNTRY.DAT /data/inventory +R
```

The first example copies the text file CHAPTER1.TXT from the FLEX working disk to the UniFLEX file "/book/chapter1". The second example copies, in raw mode, the data file INVNTRY.DAT from FLEX drive 2 to the UniFLEX file "/data/inventory".

COPYUF

The COPYUF command copies a file from a UniFLEX disk to a FLEX disk.

DESCRIPTION

The syntax of the COPYUF command is

```
+++COPYUF <UniFLEX_file_name> <FLEX_file_name> [+R]
```

where the UniFLEX file name can be any standard UniFLEX file specification. The extension for the FLEX file name defaults to TXT. The 'R' option tells COPYUF to copy the file in raw mode--that is, character-by-character without any changes. If the user does not specify the 'R' option, the UniFLEX file is assumed to be a text file, and space compression is performed on the FLEX file as it is written. All binary files and most data files should be copied with the 'R' option. Some examples of COPYUF follow:

```
+++COPYUF document DOCUMENT
+++COPYUF /bin/dir UDIR.BIN +R
```

The first example copies the UniFLEX file "document", which is found in the working directory (the directory the user was in when FLEX was called), to the FLEX working disk and calls it DOCUMENT.TXT. The second example copies, in raw mode, the "dir" command from the UniFLEX directory "/bin" to the FLEX working disk. While this process yields an exact copy of the binary code for the UniFLEX "dir" command, the "dir" command definitely does NOT run under "FLEX for UniFLEX" because the record format for a FLEX binary file differs from that of a UniFLEX binary file.

FNEWDISK

Create and format a file-disk (a UniFLEX file used as a FLEX disk). The file-disk is a regular UniFLEX file which appears to "FLEX for UniFLEX" to be a standard FLEX-formatted disk.

DESCRIPTION

The syntax of the FNEWDISK command is

```
+++FNEWDISK <UniFLEX_file_name>
```

where the <UniFLEX_file_name> is a standard UniFLEX file specification. Normally, the UniFLEX file name should be in lowercase, but it is permissable to use uppercase if desired.

As does the standard FLEX command NEWDISK, FNEWDISK prompts to be sure that formatting should take place. FNEWDISK also prompts the user for the number of tracks desired. File-disks consist of logical tracks which each contain 32 logical sectors. Since each logical sector contains 256 bytes, each track contains 8K of space. The number of tracks can range from 2 to 255, implying file-disk sizes of 16K to approximately 2 Megabytes. For example, a file-disk formatted with 40 tracks yields 320K of space.

An example of FNEWDISK follows:

```
+++FNEWDISK /flexdisk_2
```

This example creates the file-disk "flexdisk_2" in the root directory. The user is prompted for the number of tracks to establish in "flexdisk_2".

6. FLEX PROGRAM COMPATIBILITY

As previously described, almost any FLEX program which does not attempt to directly access the I/O hardware should run under "FLEX for UniFLEX". One other stipulation is that the program must be interruptable. Most programs meet both of these qualifications and present no problem for "FLEX for UniFLEX".

At the time of this writing, three of Technical Systems Consultants' FLEX-based programs have some difficulty running under "FLEX for UniFLEX". These packages are discussed in this section.

1. FLEX Diagnostics Package

The FLEX Diagnostics Package consists of two parts: a set of memory tests and a set of disk diagnostics. The memory tests call on monitor ROM routines for terminal I/O rather than calling FLEX routines, since that makes it possible to test the memory in which FLEX is residing. Therefore, the memory tests cannot be used with "FLEX for UniFLEX".

A few of the disk diagnostic programs were originally designed to work only with known, standard FLEX disk formats. These programs would not function with nonstandard formats such as the file-disks of "FLEX for UniFLEX". Recently, however, all the disk diagnostics programs were upgraded so that they work properly with nonstandard FLEX disk formats as well. The diagnostics in the new release, which are version #4, can be used with "FLEX for UniFLEX" (on either file-disks or regular disks). Older versions can be updated through Technical Systems Consultants' normal update procedure.

2. FLEX BASIC and Extended BASIC

Both FLEX BASIC and Extended BASIC run under "FLEX for UniFLEX" without modification. There is one slight difference from their normal operation, however. These BASICs normally "watch" for a control-C break from the user by directly checking the ACIA to see if the user has typed a character. They cannot, of course, directly access the ACIA under "FLEX for UniFLEX". The locations which they check for an ACIA happen to be actual memory locations, which always return values that indicate that no character was typed.

The BASICs, therefore, assume that no control-C was typed, and they continue. Thus, their mechanism for trapping control-C breaks is not functional under "FLEX for UniFLEX". This does not create a problem because the control-C key still stops the executing program. UniFLEX traps the control-C and, instead of passing it on to FLEX, instructs FLEX to jump to the address in the FLEX Return Register. This jump

produces the same result as the normal BASIC control-C break--the BASIC program is terminated and the READY prompt reappears.

3. FLEX Debug Package

At the time of this writing, the FLEX Debug Package does not function properly with "FLEX for UniFLEX" because the simulation routines are not interruptable. If the state of the interrupt masks does not matter to the program being debugged, existing versions of the debugger can be made to work by setting the interrupt mask in the debugger's condition code register before beginning simulation. The following command accomplishes this:

```
SET CC=50
```

A new release of the Debug Package, which will correct this problem, is planned. All versions of the package up to and including version #19 do not function under "FLEX for UniFLEX". When versions which correct the problem are released, they will be available through the normal Technical Systems Consultants' update procedure.