OF_LINK
=======
Version 0.5

User's Manual

Sardis Technologies
(604) 255-4485 (Pacific Time)

2261 East 11th Avenue
Vancouver, B.C.
Canada   V5N 1Z7

Printed in Canada

CREDITS

Special thanks to Will W. for his suggestions.

TRADEMARKS

COPYRIGHT NOTICE

DISCLAIMER AND WARRANTY

This manual last revised August 1, 1988 3:53 pm   rm#77   oflink.DW097

TABLE OF CONTENTS

INTRODUCTION AND SPECIFICATIONS

OF_LINK is an OS-9 module that acts as an interface between the FLEX and OS-9 operating systems. It permits FLEX to run underneath, and simultaneous with, OS-9, in a similar fashion to other systems that let MS-DOS run as a "guest" operating system under UNIX. Other features of the OF_LINK package are:

- includes a special copy utility to let you transfer files from OS-9 to FLEX, or vice versa.
- programs running under FLEX with OF_LINK can use FLEX and OS-9 system calls within the same program! This means you can write a FLEX program that uses OS-9's high-level graphics calls.
- lets you switch between OS-9 and FLEX without re-booting (using OS-9's multi-window environment)
- multiple users possible under FLEX by running 2 or more copies of FLEX simultaneously
- lets you use your CoCo 3's 80 column text display, high resolution multi-color graphics screens, RAM-Disk, hard disks, disk caching, etc. with FLEX, even though existing versions of FLEX for the CoCo didn't come with such support. No patching of FLEX is necessary.
- lets you boot FLEX from physical FLEX disks (unlinked or linked), or from a virtual disk contained in an OS-9 file. You can even boot FLEX from a hard drive!
- includes OF-MON, a mini-monitor program that takes the place of the ROM based monitors often present on other FLEX systems.
- lets you use your existing copy of FLEX or SK*DOS, even if it was not configured for the CoCo. (Refer to the next section regarding which versions are compatible with OF_LINK.)
- FLEX printer spooling is not supported in this version.

NOTE -- throughout this manual, all references to FLEX also apply to SK*DOS (STAR-DOS), unless otherwise explicitly stated.

SYSTEM PREREQUISITES -- or what you need to run OF_LINK

1) A Tandy Color Computer 3 (CoCo 3) with 512K memory. OF_LINK will not run with only 128K.

2) Radio Shack's OS-9 Level II Version 2.00.01, or later, for the CoCo 3.

3) SDISK3 disk driver module from D.P. Johnson installed on an OS-9 system disk (boot disk). While the standard version of SDISK3 available directly from D.P. Johnson (or his dealers such as Microcomm or Clearbrook Software Group) will work with OF_LINK, the special Sardis Technologies' version for the DMC floppy disk controller will work better.

4) A copy of the FLEX or SK*DOS (STAR-DOS) operating system. The versions that have so far been verified to be compatible with the OF_LINK package are:
   a) FLEX - General Version 3.01 from Technical Systems Consultants
   ~~b) FLEX - Version 9.0 (older version than 3.01) for the Southwest Technical Products Corp. DC-2 controller, from Technical Systems Consultants~~
   c) FLEX - FLEX CoCo Jr. / FLEX CoCo Sr. from SouthEast Media
   d) SK*DOS - STAR-DOS, June '85 version, from STAR-K

Many other versions should also run without changes. If your version of FLEX or SK*DOS is not on the above list and has problems running with OF_LINK, contact us and we will try to help (but no guarantees). Note - many of the versions of FLEX sold by Southwest Technical Products Corp. were so highly modified internally that they have problems working with OF_LINK.

The OF_LINK package supplies its own console, printer, and disk driver routines, so it doesn't matter if the FLEX or SK*DOS version you choose does or doesn't support the disk configuration you need.

HOW TO INSTALL OF_LINK

The following instructions assume that you have at least one floppy disk drive (48 tpi), and one hard drive. Since some people have named their hard drive as /D0, while others use /H0 for their hard drive and leave their floppy drive as /D0, our instructions below use /F0 for the floppy drive and /H0 for the hard drive to avoid confusion. Just replace the /F0 and /H0 with whatever names you use in your system for the floppy and hard drives, respectively.

If you don't have a hard drive, but have two floppies instead, replace the /F0 and /H0 with the names of your two floppy drives. First make a copy of your OS-9 system disk (boot disk). It must have SDISK3 installed and have at least 70 free sectors, but preferrably more than 140 sectors. Then for the entire duration of these installation instructions, leave this disk in the drive referred to as /H0 below. You will end up with an OS-9 format disk containing a FLEX virtual disk that is your FLEX system (boot) disk.

If you want your boot disk to be a physical FLEX floppy disk, not a virtual disk, or if you have some other hardware configuration, well, you'll just have to read the manual thoroughly until you understand the OF_LINK package, and come up with your own modified installation instructions.

[ ] Backup the OF_LINK disk we supplied (CoCo OS-9 single-sided, double-density 35 track format) to create a working copy, then put the original disk away into a safe place.

[ ] Put the working copy of the OF_LINK disk you just created into drive /F0, then type:

```
CHX /F0/CMDS
```

[ ] If you are using the Sardis Technologies' DMC version of SDISK3, skip this step and go on to the next step. If you are using the unmodified SDISK3 from D.P. Johnson, you need to patch OF_LINK before running it:

```
dpoke /F0/CMDS/of_link 0015 00
newcrc /F0/CMDS/of_link
```

[ ] Now type:

```
of_link /F0@ /F0@ /F0/VIRTUAL
```

[ ] When prompted to do so, replace the disk in /F0 with a disk containing the FLEX or SK*DOS operating system. This will be followed by a prompt for the name of the file containing the operating system. If your disk has both a .COR and .SYS version of the operating system file, the .COR version is preferred.

[ ] When FLEX or SK*DOS is booted (as indicated by the "+++" or "STAR-DOS:" or "SK*DOS:" prompt), put the OF_LINK disk back into drive /F0 and type:

```
        VFORMAT.2
        complete OS-9 pathname - /H0/FLEXVIRTUAL
         (etc.)
        file doesn't exist;  create it? - Y
```

The size of the volume (in sectors) that you specify will depend on how much free space is on the disk in /H0, and on the number and size of the FLEX programs you want on the FLEX boot disk. You must specify at least 66, but may need several hundred.

[ ] The newly formatted virtual disk can't be accessed until we re-boot, so use the "MON" and "O" commands to exit FLEX/SK*DOS and return to OS-9.

[ ] With the OF_LINK disk still in drive /F0, type:

```
        of_link /F0@ /F0@ /F0/VIRTUAL /H0/FLEXVIRTUAL
```

[ ] When prompted to do so, replace the disk in /F0 with a disk containing the FLEX or SK*DOS operating system, and also respond to the prompt for the name of the operating system file.

[ ] When FLEX or SK*DOS is booted, copy all desired files from the FLEX or SK*DOS physical disk in drive /F0 to the new virtual disk. For example, to copy all files, type:

```
        COPY 0,3
```

However, if the virtual volume is not large enough to contain all the files, you will need to selectively copy only the most important ones, including of course, the file containing the operating system itself. For example:

```
        COPY 0,3 flex.cor,copy,delete,rename,cat,date,ttyset,asn,build
```

There are several files you do NOT want to copy over, as they are not compatible with the OF_LINK environment. If you copied ALL files, delete these incompatible files from drive #3. Some of them are:

```
        NEWDISK.CMD, FORMAT.CMD, P.CMD, PRINT.SYS, PRINT.CMD, QCHECK.CMD,
         LINK.CMD
```

If your version of FLEX or STAR-DOS was originally configured for the CoCo, there will be many other files that cannot be run under OF_LINK without crashing the system. We can't list all the files for all the possible versions of FLEX you might have, so you will have to come up with your own list, but some of them will have names such as:

```
        CCBASIC.CMD, DISKRATE.CMD, MAKESYS.CMD, RSDIR.CMD, RSREAD.CMD,
        RSWRITE.CMD, SAVEROM.CMD, TERM.CMD, TESTDISK.CMD, USERKEYS.CMD,
        V51.CMD, V32.CMD
```

[ ] Now put the OF_LINK disk into drive /F0 and copy all the new OF_LINK specific files to the new virtual disk by typing:

```
        COPY.3 2,3
```

[ ] If you want to avoid typing the name of the operating system file every time you boot, type the following command, where ffffffff.eee is the name of the operating system file (eg. FLEX.COR):

```
        LINK.3 3.ffffffff.eee
```

[ ] Use BUILD to create a customized STARTUP file on the virtual disk, for example:

```
        +++BUILD.3 3.startup.txt
        =dskset dr=2,tm=96:ttyset be=08:asn w=1:date:dskshow
        =#
```

[ ] Use the "MON" and "O" commands to exit FLEX/SK*DOS and return to OS-9, as we will want to patch OF_LINK for our new customized configuration.

[ ] To tell OF_LINK that we want FLEX's drive #0 (the boot drive) to be the virtual drive, and to omit the prompt for the operating system file, type the following sequence (with the OF_LINK disk still in drive /F0):

```
        dpoke /F0/CMDS/of_link 0018 01
        dpoke /F0/CMDS/of_link 001C 00
        newcrc /F0/CMDS/of_link
```

The above patches will set FLEX's drives 0, 2, and 3 as virtual drives, and leave drive 1 as a physical drive. Refer to the write-up on the OS-9 module "OF_LINK" (later in this manual) for details if you want some other configuration.

[ ] Copy the patched OF_LINK module into the CMDS library on your OS-9 system disk:

```
        copy /F0/CMDS/of_link /H0/CMDS/of_link
```

Remove the OF_LINK disk from drive /F0 and put it away.

[ ] Now you are ready to boot FLEX or SK*DOS in its configured state from the virtual disk, and the following command is an example of what you will key in the future every time you boot:

```
        CHX /H0/CMDS
        of_link /H0/FLEXVIRTUAL /F0@ P=/P
```

NEW FLEX UTILITY COMMANDS

The next few pages describe additional commands, supplied by Sardis Technologies, included on the OF_LINK disk, that run under FLEX or SK*DOS in the OF_LINK environment. They are contained in the FLEX virtual volume on the OS-9 format disk included as part of the OF_LINK package.

NOTE -- in the syntax descriptions of each command, the commas shown separating the parameters from each other and from the command name can be replaced by spaces (blanks).

BINCPY BINCPY

The BINCPY command copies FLEX binary files to compact them by squeezing out wasted space and optionally allowing the user to omit some binary records. After using the FLEX APPEND utility to add small patches to programs, or to the FLEX operating system file itself, the binary file usually contains hundreds of bytes of null data interspersed throughout the file. By packing the binary records and omitting the null bytes, BINCPY can often reduce the size of the binary file by one or more sectors. The syntax of BINCPY is:

```
     BINCPY <input filename>,<output filename>[,P]
```

The file extension for both files defaults to ".BIN". The optional "P" parameter at the end puts you into a prompted copy mode which lets you display each binary record and decide to copy or omit that record. If the "P" parameter is left off, BINCPY simply copies and packs the entire binary file without any prompting.

In the prompted mode, BINCPY displays the starting address, ending address, and size of a binary record, then prompts you to choose one of 5 possible options for that record:

C - copy the record shown, then go to the next record
N - no, don't copy the record shown (ie. omit it), but go to the next record
D - display the contents (in hex and ASCII) of the record shown, then reprompt for the same record
E - immediately exit the BINCPY program; only those binary records copied before this response will appear in the output file
H - display help menu

If you press a wrong key, you will be reprompted for the same binary record. Both kinds of binary records are shown, data records, and transfer address records. Note that the "D" option is invalid for a transfer address record, as there is no data in such a record other than the address which is already shown in the prompt.

If P.CMD precedes BINCPY on the command line, only the record display ("D" response to the prompt) is sent to the printer; all other prompts and displays appear on the console.

The selective copying feature can be used to extract portions of a program, which can then be disassembled, modified, re-assembled, and then merged back into the original program. With this procedure the patched program takes up less space than if patch overlays had been merely appended to the original program. Another use for the selective copying would be to strip off I/O driver routines from a FLEX.SYS file to create a FLEX.COR file.

For more information on FLEX binary records, refer to the "FLEX Advanced Programmer's Guide", or to the "SK*DOS User's Manual".

Examples:

```
  1) RENAME 0.FLEX.SYS,FLEX.OLD
     BINCPY 0.FLEX.OLD,0.FLEX.SYS
     LINK 0.FLEX.SYS

  2) P BINCPY TEST JUNK P
```

In the first example, BINCPY is used to squeeze out wasted space in the FLEX.SYS file, then the new file is LINK'd to for booting. In the second example, the P.CMD output override is specified immediately before "BINCPY" so selected records from the TEST.BIN file on the working drive can be dumped to the printer. The JUNK.BIN file on the working drive will only be created if records are selected to be copied. The "P" parameter at the end of the line selects the prompted mode.

The BINCPY program and documentation have been adapted from the public domain program of the same name written by J.C. Hausler that appeared in '68' Micro Journal December '84 pp.41-44.

BLIST BLIST

===

The BLIST command is similar to the LIST command, in that it lists the contents of text files to the screen or printer. However, it uses a very large buffer to reduce wear and tear on the disk drives, especially valuable when a slow printer is being used. Pageination and line numbers are not supported in this version. Its syntax is:

```
    BLIST,<file spec>
```

where <file spec> is the name of the text file to list. An extension of .OUT is assumed unless otherwise specified. BLIST may be preceded with the "P" command. The "escape character" feature of FLEX (refer to TTYSET) is active. BLIST and its buffer occupy most of $0100-$7FFF of user memory while running.

CHECKSUM

The CHECKSUM command reads any disk file and computes a checksum on it. This is a quick way of determining whether or not two copies of a program or text file are identical versions.

Another way of using it is, every time you make changes to a program or other file, to calculate the checksum of the new version and write it in a log. Later, if you want to determine which version a particular copy is, calculate the checksum on it and look it up in the log.

The syntax of the command is:

```
    CHECKSUM,<file spec>
```

where <file spec> is the name of the file on which to calculate the checksum (the extension defaults to .BIN). The 2 byte checksum is displayed as a four digit hexadecimal number.

Another possible use for this command is to simply read a file to see if any sectors in it have read errors.

DSKSET

The DSKSET utility lets you change various disk parameters without having to resort to machine language patching. You can view the current setting of these parameters with the DSKSHOW utility that is described elsewhere. The syntax of DSKSET is:

```
   DSKSET,<parameter list>
```

where <parameter list> is a list of 2 character parameter codes, each followed by an equals sign "=" and by the value being assigned. Each parameter should be separated from the next one by a comma or space (blank). All parameters following a DR=d parameter apply only to that specified drive, until the next DR=d parameter. If no DR parameter is specified, drive 0 is assumed. Some examples are:

```
  +++DSKSET TM=96
  +++DSKSET DR=1,SS=9,SD=17
```

The first example sets drive number 0's media track density flag to "96 tpi". The second example changes the number of sectors per side, for drive 1, to 9 and 17 for single and double density, respectively.

Note -- PFORMAT also updates some of these values. More specifically, the media values for the drive containing the disk being formatted are updated as per the responses to PFORMAT's prompts re double/single density, lower density Y/N. Also, since PFORMAT doesn't handle "double stepping", the TM (media track density) flag is set to the same tpi (tracks per inch) value as the drive itself.

The DSKSET command is serially re-usable and position independent, so could be made memory resident, if desired.

Here is a list and description of all DSKSET parameters:

DR=d   select DRive

Specifies to which drive the following parameters apply, where "d" is 0, 1, 2, or 3.

TM=48 or TM=96   Track density of Media (tpi)

Specifies whether the diskette currently inserted is formatted with 48 or 96 tpi (tracks per inch).

SS=dd   Sectors per side of Single density track

This parameter specifies how many sectors are on one side of one single density track of the diskette currently inserted. Valid values are 9 or 10. The TSC standard is 10. If this value does not match the actual format of the diskette in that drive, the system will look for some sectors on the wrong side of the disk and will never find them.

SD=dd   Sectors per side of Double density track

This parameter specifies how many sectors are on one side of one double density track of the diskette currently inserted. Valid values are 16, 17, or 18. The TSC standard is 18. If this value does not match the actual format of the diskette in that drive, the system will look for some sectors on the wrong side of the disk and will never find them.

DSKSHOW   DSKSHOW

===

The DSKSHOW utility lets you view the current settings of several disk parameters. Refer to the description of the DSKSET utility above on how to change some of these parameters. The syntax of DSKSHOW is simply:

```
   DSKSHOW
```

The DSKSHOW command is serially re-usable and position independent, so could be made memory resident, if desired.

Here is a list and description of all DSKSHOW parameters:

DR   DRive

Specifies the logical FLEX drive number for the drive displayed in each of four columns.

      OS-9 Pathnumber

These values are the OS-9 path numbers used by OF_LINK for each logical FLEX drive and remain unchanged for the duration of a session (until you reboot).

DV   Type of DriVer

The codes shown indicate whether the FLEX logical drive is associated with a physical disk (P), or is a virtual disk (V) in an OS-9 file. Refer to the description of the OF_LINK module later in this manual on how to change these codes.

      Double Sided drive

This indicates whether the disk drive itself is capable of double-sided operation (2 sides), or not (1 side). This value is copied from the corresponding OS-9 device descriptor, and cannot be changed by DSKSET. (not shown for virtual disks)

      Track density of Drive (48/96 tpi)

Specifies whether the drive steps at 48 or 96 tpi (tracks per inch). This value is copied from the corresponding OS-9 device descriptor, and cannot be changed by DSKSET. (not shown for virtual disks)

TM   Track density of Media (48/96 tpi)

Specifies whether the diskette currently inserted is formatted with 48 or 96 tpi (tracks per inch). You can change this value with DSKSET. PFORMAT also updates it, to the same value as the "track density of drive" above. (not shown for virtual disks)

      Density

The value shown here is the system's latest guess as to whether the disk is single (S) or double (D) density. Since the disk drivers automatically determine this, you can't use DSKSET to change this value. Note that When a new diskette is inserted, this value will not change until a track other than track 0 is read or written. PFORMAT also updates this value as per the response to the prompt "double density?". (not shown for virtual disks)

SS   Sectors per side of Single density track

This parameter specifies how many sectors are on one side of one single density track of the diskette currently inserted and can be updated using DSKSET. Valid values are 9 or 10. The TSC standard is 10. If this value does not match the actual format of the diskette in that drive, the system will look for some sectors on the wrong side of the disk and will never find them. PFORMAT sets this value to 10 unless "lower density" is selected, in which case it sets it to 9. (not shown for virtual disks)

SD   Sectors per side of Double density track

This parameter specifies how many sectors are on one side of one double density track of the diskette currently inserted and can be updated using

DSKSET. Valid values are 16, 17, or 18. The TSC standard is 18. If this value does not match the actual format of the diskette in that drive, the system will look for some sectors on the wrong side of the disk and will never find them. PFORMAT sets this value to 18 unless "lower density" is selected, in which case it sets it to 16. (not shown for virtual disks)

OS-9 Pathlists Used

The OS-9 pathlists you told OF_LINK to use for each FLEX logical drive are shown here, and remain unchanged for the duration of a session (until you reboot).

A sample display is:

```
    DSKSHOW

 Drive Characteristics       DR       0     1     2     3
                                     --    --    --    --
  OS-9 path number                    3     4     5
  type of driver             DV       P     P     V     V
  sides                               2     1
  48/96 tpi                          48    96

 Media currently inserted

  48/96 tpi                  TM      48    96
  density (S/D)                       S     D
  sectors/side (SD track)    SS      10     9
  sectors/side (DD track)    SD      18    16

 OS-9 pathlists used

  drive 0 = /d0@
  drive 1 = /d2@
  drive 2 = /h0/FLEX/virtual4
```

LINK   LINK

The LINK command is used to update a system (boot) disk with a pointer to the FLEX or SK*DOS file that contains the operating system itself (or other program) to be loaded when booting the system. If you have "LINK'd" a system disk, OF_LINK can determine which file to load without prompting you for the file name.

Most versions of FLEX or SK*DOS already come with the LINK command, and you can use either ours or theirs. We supply our own because a few versions omitted this command.

NOTE - any time the file that was LINK'd to is moved, either because you created a new version with additional code using APPEND, or COPY moved it, you need to re-run LINK to update the pointer to the new location of the file.

LOADO

The LOADO command is used to load a binary file from disk into memory. An optional address offset lets you load the program into a different location than specified by the file. Its syntax is:

```
    LOADO,<file spec>[,<offset>]
```

where <file spec> is the name of a binary file to be loaded (extension defaults to .BIN), and <offset> is an optional 4 digit hexadecimal address displacement. Refer to the FLEX Programmer's Manual for a description of the offset field (location $CC1B-$CC1C) and the LOAD routine (address $CD30), as these are used by LOADO.

Some examples are:

```
+++LOADO ABCDE.CMD
+++LOADO WXYZ.1,E000
```

The first example loads binary file "ABCDE.CMD" from the work drive into those memory locations specified in the file. The second example loads binary file "WXYZ.BIN" from drive 1 into memory, but at locations $2000 lower than specified. (Eg., a byte destined for $9374 is loaded into $7374 instead, because $9374 + $E000 = $7374 when the result is truncated to 16 bits.)

Note -- LOADO resides in FLEX's Utility Command Space (starting at $C100); don't use it to load another program into this area.

MON

MON is used to temporarily exit FLEX and execute the OF-MON monitor program which is described elsewhere.

To re-enter FLEX after using the MON command, use OF-MON's "FW" or "FU" commands; do NOT use OF-MON's "J" command to directly jump to $CD03 as you do on most other systems.

P

The P command (P.CMD) allows you to redirect output from the screen to the printer instead. It has the same basic function as the P.CMD supplied with FLEX or SK*DOS itself. However, many of those versions of P cannot be used with OF_LINK, as they include code that accesses the printer port hardware directly. Our P.CMD neither includes any hardware specific code, nor does it attempt to load a PRINT.SYS file, as our printer driver is built-in to OF_LINK itself.

Also refer to the description below of the PRINT.SYS file.

PFORMAT

PFORMAT is a replacement for the FLEX "NEWDISK" command. It is used to format actual (physical) floppy disks. Refer to VFORMAT for formatting virtual disks. The syntax used by PFORMAT is:

```
  PFORMAT,<drive>
```

where <drive> is a single digit number (from 0 to 3) specifying which drive will contain the disk to be formatted. PFORMAT issues a series of prompts before beginning the actual formatting process, so the same drive can be used to load and execute PFORMAT from disk, as will be used to hold the disk being formatted.

Most of the prompts are self-explanatory and are answered with either a "Y" or "N" (refer to the sample session below). At the end of the prompting session, but before formatting begins, you will be given another chance to abort or continue. Several prompts that need more explanation are:

a) LOWER-DENSITY FORMAT? - Usually you will respond with "N" to select the TSC standard of 10 sectors per track for single density and 18 sectors per track for double density. If you respond with "Y", you will get fewer sectors per track (9 for single, 16 for double density) with longer gaps between each, similar to the IBM 3740 standard.
b) SWTPC FORMAT? - This prompt only has significance if the disk is to be read or written on another system. Most systems that use Western Digital controller chips numbered 1791, 1793, 2791, 2793, 1770, 1772, or 1773 (such as on the Sardis Technologies' DMC controller) can read disks formatted with a response of "Y" as well as those formatted with "N" (as long as the disk driver routines disable the side flag compare). Systems using the 1795, 1797, 2795 or 2797 chips are more fussy. Most manufacturers using these latter chips properly implemented the side flag -- you must use a response of "N" for disks to be read on their systems. But a few, such as Southwest Technical Products Corp. (SWTPc), used the side flag to select single vs double density -- you must respond with "Y" to the prompt in order for disks to be able to be read or written on their systems.
c) VOLUME NAME - the name must follow standard FLEX file name rules.
d) VOLUME NUMBER - any number from 0 through 65535.

After the last prompt, the formatting process will take approximately 45 seconds for a single sided, single density 35 track disk, to approx. 1.5 minutes for double sided, double density 40 tracks. Defective sectors will slow down the process. Any sector that is deemed "bad" will be reported as "BAD SECTOR AT TT SS" where "TT SS" is the track/sector number in hexadecimal.

If there any bad sectors on track zero, or if the very first or very last sector in the data chain is bad, or if 50 or more sectors have errors (soft or hard) in the verify pass, the formatting will be aborted.

If you have many errors when formatting a disk, here are several possible causes you should check out:

- the drive's rotational speed might not be within specifications; use the RUN "*" program on the OS-9 BOOT/CONFIG disk to check it.
- the track at which write precompensation is activated may not be optimal; if you are running OF_Link on a system that uses the Sardis Technologies' DMC controller, study the "Changing User Values" section in the SDISK-DMC Software manual regarding the write precompensation table.
- using a bulk tape eraser on the disk being formatted often helps (but be careful not to accidently erase other disks!)

NOTE:

- PFORMAT erases the contents of FLEX's command line buffer, so any commands specified after PFORMAT, on the same command line, will be ignored.
- PFORMAT and its track buffers occupy most of $0100 - $7FFF of user memory while running.
- PFORMAT does not support double stepping, so 96 tpi drives will format disks only at 96 tpi.
- only 35, 40, or 80 tracks can be formatted with the current version.

A sample session would look like this:

```
+++PFORMAT 1
PFORMAT - (c) 1988 by Sardis Technologies
Are you sure you want to format a physical disk? (Y/N) - Y
35 tracks? (Y/N) - N
40 tracks? (Y/N) - N
80 tracks? (Y/N) -Y
Lower-density format? (Y/N) - N
Double-density? (Y/N) - Y
Double-sided? (Y/N) - Y
SWTPC format? (Y/N) - N
Volume name - TEST
Volume number - 1
Insert disk to be formatted into FLEX's drive 1
 (C)ontinue, or (A)bort - C
Write pass in progress ....
Write pass done, verify pass in progress ....
Verify pass done
Formatting completed OK
Sectors formatted = 2844
```

PRINT.SYS   PRINT.SYS

This is a dummy file that doesn't do anything, but is provided merely so those packages that insist on finding a file with this name in order to run can run under OF_LINK. Refer to the P.CMD description above.

VFORMAT   VFORMAT

VFORMAT is a replacement for the FLEX "NEWDISK" command. It is used to format a FLEX virtual disk that is contained within an OS-9 file. Refer to PFORMAT for formatting physical (real) disks. The syntax for VFORMAT is simply:

```
      VFORMAT
```

All necessary parameters are obtained by prompting. VFORMAT can format a virtual volume even if it is not currently associated with any FLEX logical drive number. The OS-9 file holding the FLEX virtual volume may already exist, or will be created if it doesn't exist yet. VFORMAT issues a series of prompts before beginning the actual formatting process, so the same disk drive can be used to load and execute VFORMAT from disk, as will be used to hold the disk being formatted.

Most of the prompts are self-explanatory and are answered with either "Y" or "N" (refer to the sample session shown below). At the end of the prompting session, but before formatting begins, you will be given another chance to abort or continue. Several prompts that need more explanation are:

a) "Complete OS-9 pathname" - specify the complete pathlist (begins with the "/") of the OS-9 file that is to contain the FLEX virtual volume.
b) "Volume size (in sectors)" - the volume can range in size from 66 to 8130 sectors (of 256 bytes each), for 16K to 2 Megabytes of storage. The number you specify is internally rounded up to the nearest multiple of 64, then has 2 added to it. This is because virtual disks have 64 sectors per "track" on tracks 1-127, plus 2 sectors on "track" 0 (system information record, and first directory sector).
c) "Directory size (number of files)" - this is just your guess of how many files, at most, this volume will ever contain. VFORMAT uses this number to create an initial directory of the appropriate size. If you guessed too low, FLEX will later expand the directory size automatically, but performance will then be reduced, since the additional directory sectors could be scattered all over the volume.
d) "Volume name" - the name must follow standard FLEX file name rules
e) "Volume number" - any number from 0 through 65535

Note - VFORMAT erases the contents of FLEX's command line buffer, so commands specified after VFORMAT, on the same command line, will be ignored.

A sample session would look like this:

```
+++VFORMAT
VFORMAT - formats a virtual FLEX volume in an OS-9 file
  (c) 1988 by Sardis Technologies
Complete OS-9 pathname - /HO/FLEX/virtual3
Volume size (in sectors) - 2500
Directory size (number of files) - 150
Volume name - VIRTO3
Volume number - 1
 (P)roceed with formatting, (R)especify, or (A)bort - P
File doesn't exist;  create it? (Y/N) - Y
Formatting of virtual disk in progress ....
Formatting completed OK
```

XCOPY

The XCOPY command is used to copy the contents of one file from a FLEX format disk to an OS-9 format disk, or vice versa. Both ASCII text and non-text (eg. binary) files can be copied, although binary files are only copied "raw", ie. they are not converted into the native load format of the destination operating system. The syntax of XCOPY is:

```
     XCOPY,<origin file spec>[,<origin options>],
              <destination file spec>[,<destination options>]
```

The first file specification is for the "source" or "from" file, followed by its option parameters, and the second file specification is for the "to" or "destination" file, followed by its option parameters. FLEX or OS-9 files can be specified for each, allowing for four possible transfer combinations: FLEX to OS-9, OS-9 to FLEX, FLEX to FLEX, OS-9 to OS-9. Since it's not always possible to determine from a file name alone whether it represents a FLEX or OS-9 file, FLEX file names must be preceded with "F=" and OS-9 path lists with "O=". Either spaces or commas may be used to separate file names and options from each other.

The only option currently implemented is the "-S" option for FLEX files. Unless you specify otherwise, XCOPY assumes FLEX files are ASCII text files, so uses an extension of ".TXT" and enables automatic space compression and expansion. If the FLEX file being read was not created with automatic space compression (eg. it's a binary file) and/or the FLEX file being written is not to have spaces compressed, use the -S option after the appropriate file spec(s).

Examples:

```
  1) XCOPY,F=oldfle,F=newfile.2,-s
  2) XCOPY O=/HO/SOURCE/myprog.txt F=1.yourprog.bak
  3) XCOPY f=junk.out.0 o=gold
```

Example #1 copies and converts the compressed FLEX text file OLDFILE.TXT on the working drive to a new uncompressed FLEX text file NEWFILE.TXT on drive 2. Example #2 copies the OS-9 text file MYPROG.TXT in directory /HO/SOURCE to a new (automatically space compressed) FLEX text file YOURPROG.BAK on drive 1. Example #3 copies the space compressed FLEX text file JUNK.OUT on drive 0 to the new OS-9 file GOLD in OS-9's current working directory, with the compressed spaces automaticaly expanded.

For more details regarding FLEX's automatic space compression/expansion, refer to the "FLEX Advanced Programmer's Guide" or "SK*DOS User's Manual".

Since XCOPY uses both FLEX and OS-9 system calls, any error messages are preceded with "FLEX ERROR" or "OS-9 ERROR" so you can look up the messages in the appropriate manuals.

CHANGES TO TSC'S FLEX MANUALS AND STAR-K'S SK*DOS MANUALS

1) The FLEX manuals state that, after using the MON command, to re-enter FLEX by jumping to location $CD03. Do NOT do this with OF_LINK. Refer to the descriptions of the "J", "F W", and "F U hhhh" commands in the OF-MON section elsewhere in this manual.

2) FLEX manuals talk about "automatic drive searching" in several places:
   a) the description of the "ASN" command in the FLEX User's Manual
   b) the description of FMS function 20 "Find Next Drive" in the FLEX Programmer's Manual
   c) the description of the "Check Drive Ready" disk driver routine in the FLEX Programmer's Manual

   An OF_LINK system, with 5" drives attached, functions (more or less) like TSC's description of a system with 8" drives. All four drives will be searched, and the system will not hang up if a drive is empty.
   (>>> not implemented yet <<<)

3) If automatic drive searching is not invoked (ie. ASN has set the system and/or work drive to a specific drive) or an explicit drive number is specified in a command, operation is slightly different. Attempting to to access a drive not containing a disk will NOT hang up the system forever (or until a disk is inserted and the door closed) as on most other 5" systems, so do NOT press the reset button. However, the system will take as long as one minute before timing out with an error message, so be patient!

4) There are three routines defined in newer versions of FLEX that were missing in the earliest versions:
   a) $D3E5-$D3E6 contains an address pointing to a routine that inputs a character from the console keyboard without echoing it. It was the omission of this routine that caused the present "mess" with so many FLEX software packages requiring patching to run on systems that don't have a 6850 ACIA at $E004. OF_LINK goes one step further and adds a "JMP" ($7E) opcode at $D3E4 so you can use either direct jumps to $D3E4 or indirect jumps using $D3E5 to use the input character routine.
   b) $DE18-$DE1A has a jump instruction to a "warm start" disk driver subroutine that is called during the FLEX warmstart procedure.
   c) $DE1B-$DE1D has a jump instruction to a "SEEK to track" disk driver subroutine.

   OF_LINK supports all three routines, even for old versions of FLEX. Note however, if you are using one of these early versions of FLEX that the "warmstart" disk driver routine will never be called by FLEX.

5) The disk driver jump table at $DE00-$DE1D must never be changed when using OF_LINK. On the other hand, you should never have a need to do so, as drivers for other devices (hard drive, RAM-Disk, etc.) are implemented in OS-9, and used via the OF_LINK "virtual" drive capability.

6) Do NOT use the following commands or files provided with FLEX or SK*DOS:
   NEWDISK.CMD, FORMAT.CMD, P.CMD, PRINT.SYS, PRINT.CMD, QCHECK.CMD

OS-9 UTILITIES

The next few pages describe additional commands supplied by Sardis Technologies to run under OS-9.

CMDGEN3

Syntax: cmdgen3 <command name>

Function: Creates an OS-9 module which can execute a program or series of programs, with run time parameter substitution. Also displays the command sequence contained in a generated command module. command module.

Parameters:

command name	The name of the executable module to be generated or displayed. It is also the name of the file in the current execution directory containing that module. The name may be up to 29 characters long.

Notes:

* Shell procedure files can save you lots of time and keying effort when- ever you need to repeatedly run the exact same complicated command or series of commands. But procedure files lose much of their value when the command sequences need to be slightly different each time they are run, as you either first have to use a text editor to modify the file before each run, or are forced to create many versions of that proced- ure. Also, procedure files are just ASCII text files, not OS-9 modules, so they can't be loaded into memory, but must be read from disk as they are being executed.

  CMDGEN3 solves these problems by allowing the changeable parts of the command sequences to be specified at run time. It also "encapsulates" the ASCII text inside a module so it can be made memory resident.

  CMDGEN3 creates "macro" commands. That is, it lets you quickly and easily build your own custom commands from one or many other existing commands. You don't have to be a programmer, because instead of using an assembler or high level language compiler, CMDGEN3 simply lets you key in the sequence of commands as you would if you were running them directly from the shell. New commands can be generated so quickly they can even be used for "throw away" commands that you generate, use, and discard, all in a few minutes.

  Like regular OS-9 commands, your new command can accept parameters (up to three) which it in turn passes on to the commands it is composed of. This parameter substitution facility is especially useful when the same changeable value is used repeatedly within a command sequence, such as in example #5 below.

* To create a new macro command, run CMDGEN3 and specify the name of the command module to be created. The module will be stored in a file created in the current execution directory, so you may need to first run the CHX command. When you see the "Cmdgen3:" prompt, enter the command line (up to 150 characters long) that the generated command is to execute.

  This command line may include all the features that you use in a normal shell command line (ie. the line you key after the "OS9:" prompt), such as the ";" or "!" or "&" characters to put more than one command on one line. In addition, you can insert parameter variables that are later replaced with the user supplied values when the generated command is run.

  Up to three parameter variables can be specified. Wherever you will want the first run time parameter to appear, insert the parameter variable "%1". Do not use a space before the "%" or after the "1" unless you want the space there at run time. Similarly, parameter variables "%2" and "%3" are used for the second and third run time parameter values. Each parameter variable may appear more than once in the same line, and they may be placed in any sequence. Study the examples below.

  If the "%" character is immediately followed by any character other than the digits "1", "2", or "3", it will not be interpreted as a parameter value, but will be left in the command line unchanged.

* To run the generated command, type the command module name, followed by the run time parameter substitution values (up to three). These parameters are separated from each other by blanks, so may not contain embedded blanks. Most printable characters, including commas, are allowed in a parameter, except for the eight special characters ";!<>()#&" that the shell uses for itself. Refer to the examples below. There is no limitation on the length of each parameter except for the 200 total characters maximum described below.

  The expanded command line (ie. after parameter substitutions) will be displayed, but will not be executed until you press a key (any key). This gives you a chance to inspect the command line that will actually be executed, and let you abort it with the BREAK key if it's not what you wanted to do.

  The expanded command line (after parameter substitutions) may not be more than 200 characters long -- if it is, "ERROR #70" will be displayed and the command line will not be executed.

* If you can't remember what command sequences make up an existing generated command, there are two ways to display the embedded command line. One is to run CMDGEN3 and specify the command module name. CMDGEN3 will look for it in the current execution directory, so you may need to run the CHX command first. See example #1 below. If the module is not a valid CMDGEN3 module, the message "**** Not a CMDGEN3 module!" will be displayed.

The other way is to run the generated command itself, but specify "%1 %2 %3" as the run time parameter line. When the expanded command line is displayed, it will look exactly like the original internal command line. Press the BREAK key to abort without executing it.

* Although CMDGEN3 was written by Sardis Technologies, it incorporates some code from Stephen Goldberg's original CMDGEN program, with his express permission. (See RAINBOW magazine May '88 pp. 185-189)

Examples:
----

1) The generated command module COPYSRC already exists, and we want to see what command line it contains, without executing it:

```
     OS9:cmdgen3 copysrc
     copy %1/source/%3 %2/source/%3 #24k
   or
     OS9:copysrc %1 %2 %3
     copy %1/source/%3 %2/source/%3 #24k
     <BREAK>
```

2) To create a command module that copies all files from one directory to another directory:

```
     OS9:cmdgen3 copyall
     Cmdgen3: chd %1;dsave -s16 %1 %2 #8 ! (chd %2)
```

3) This example runs the command module displayed in example #1 to copy file "junk.txt" from the SOURCE directory on drive /D1 to the SOURCE directory on /D0. Note that since you only have to type the file name once, and that it is at the end of the line you key, how easy it would be to use the control-A and backspace keys to repeat the same command for other file names:

```
     OS9:copysrc /D1 /D0 junk.txt
     copy /D1/source/junk.txt /D0/source/junk.txt #24k
```

4) To run the command module created in example #2 to copy all files from a disk in drive /D1 to a directory TEST on /R0:

```
     OS9:copyall /D1 /R0/TEST
     chd /D1;dsave -s16 /D1 /R0/TEST #8 ! (chd /R0/TEST)
```

5) To create a command sequence to edit a text file with first deleting the previous version (.BAK extension), renaming the current version (.TXT extension) to a .BAK extension, then editing the .BAK version into the new .TXT version:

```
     OS9:cmdgen3 ed3
     Cmdgen3: chd /D1/source;del %1.bak;rename %1.txt %1.bak;edit
      %1.bak %1.txt #16k
```

To run this command, merely type:

```
     OS9:ed3 essay
```

which automatically expands into a long command line that you no longer need to type in yourself:

```
chd /D1/source;del essay.bak;rename essay.txt essay.bak;edit
 essay.bak essay.txt #16k
```

DPOKE

```
DPOKE                                                                  DPOKE
============================================================================
```

Syntax: dpoke <pathlist> <offset> <data>[...]

Function: Updates a portion of a file on disk with new values that are
--------   specified on the command line.

Parameters:
----

pathlist	The name of the disk file to be updated (patched).
offset	The 16 bit offset from the beginning of the file where the string of new data is to begin writing. Specified as a 4 digit hexadecimal number (leading zeros may be omitted).
data	A list of one to 100 8-bit values that are to be written to the file starting at the offset. Specified as 2 digit hexadecimal numbers (leading zeros may be omitted). The values are separated from each other by spaces or commas.

Notes:
-----

* There are times when you need to patch a module before you load it into memory, especially if that module will be incorporated into the OS9BOOT file. OS-9 has several commands available to let you patch a module in memory (DEBUG and MODPATCH), display a module in memory (DEBUG), or display a module on disk (DUMP), but no facility to patch a module directly on disk. DPOKE provides this missing link.

  While there are more powerful disk patchers available from other sources (such as Doug DeMartinis' "dEd" on CompuServe), we needed a simple, non-interactive, command line oriented patcher for you to use in the OF_LINK installation procedure. That's why we wrote DPOKE.

* DPOKE writes the new data over the existing data in the file, but only those bytes specifically patched are altered; the rest of the file is not changed.

* After patching a module on disk, you must always update the header parity and CRC bytes using OS-9's VERIFY or our NEWCRC command.

* Don't use DPOKE unless you know what you are doing, and remember to have a backup copy of any module you patch in case you make a mistake. To check if the patch was made as desired, you can use DUMP to display the

patched module, or use CMP to compare the patched version to the un-
modified backup copy.

Examples:
----

1) The following example patches the OF_LINK module on disk to change the flag regarding prompting for the system (boot) file name to suppress the prompt:

```
OS9:dpoke /d1/cmds/of_link 001c 00
OS9:newcrc /d1/cmds/of_link
```

2) The following example uses DPOKE to change the volume name of a disk to "RT36" by directly patching the Identification Sector (LSN 0). Since this sector does not contain a module, NEWCRC should not be run afterwards.

```
OS9:dpoke /d0@ 01f 52 54 33 b6
```

NEWCRC   NEWCRC

===

Syntax: newcrc <pathlist>

Function: Recalculates and updates the header parity and CRC bytes of one or
--------   more OS-9 modules in a file on disk.

Parameters:
----

pathlist	The name of the file containing the module(s) to be updated.

Notes:
-----

* Unlike OS-9's VERIFY command, NEWCRC does not create a new file - the existing file is updated in place.

* If the file contains more than one module, NEWCRC will update each of them in turn, stopping only when it encounters the end of the file, or finds a non-module data area, whichever comes first. Finding a non-module area will generate an E$BMID error (#205).

Examples:
----

1) This example updates the header parity and CRC bytes of the OF_LINK module on disk:

```
OS9:newcrc /d1/cmds/of_link
```

OF_LINK

==

Syntax: of_link physpath [physpath [virtpath [virtpath]]] [p=prtpath]

Function: Enables the FLEX (or SK*DOS) operating system to run as a "guest" operating system under OS-9. Acts as an interface between FLEX and OS-9, boots the FLEX operating system, and provides the OF-MON monitor program for the FLEX environment.

Parameters:

physpath	The device name of a floppy disk drive, with a "@" suffix to designate raw physical I/O.
virtpath	The pathlist of an OS-9 file that has previously been formatted with VFORMAT to create a virtual FLEX volume in that OS-9 file.
prtpath	The physical device name of the printer (must be the last parameter.

Notes:

* The first non-printer path specified is for FLEX logical drive #0, the 2nd path, if specified, is for drive #1, the 3rd path for #2, and the 4th path is for logical drive #3. As initially provided, OF_LINK expects FLEX drives 0 and 1 to be physical drives, and drives 2 and 3 to be virtual drives, as indicated in the syntax listing above. However, OF_LINK can be patched so any drive can be a physical or virtual drive, as desired.

  At an offset of $0018 - $001B from the beginning of the OF_LINK module is a table of 4 bytes, one byte for each FLEX drive (0 - 3), where a $00 value flags that drive as a physical drive, $01 as a virtual drive. To change the default settings, you can either patch OF_LINK on disk, or patch it in memory every time you run it by first LOAD'ing it into memory, then using MODPATCH to change the flags before running OF_LINK.

  For example, to change FLEX drive #1 to virtual at run time, type:
```
       LOAD OF_LINK
       MODPATCH
       L OF_LINK
       C 19 00 01
       V
       OF_LINK /D0@ /D2/FLEX44 P=/P
```

  To patch it on disk:
```
       DPOKE /D1/CMDS/OF_LINK 0019 01
       NEWCRC /D1/CMDS/OF_LINK
```

* OF_LINK always boots from FLEX logical drive #0, so the path you specify for that drive must contain the FLEX operating system. It is possible to boot FLEX from a virtual disk, as well as from a physical disk. You can even boot FLEX from a hard disk if drive #0 is specified to be a virtual

disk on the hard drive.

* OF_LINK, as initially provided, always prompts you to key in the name of the file containing the operating system to be loaded (FLEX.COR, FLEX.SYS, STAR-DOS.COR, etc.) If FLEX's LINK command has been run on the boot disk, OF_LINK can find the operating system file by itself. However, you need to patch OF_LINK (on disk or in memory) to suppress the name prompt. Change offset $001C from $01 to $00 to do this.

* While OF_LINK can run with either the standard D.P. Johnson version of the SDISK3 driver or Sardis Technologies' modified SDISK3-DMC version, OF_LINK as distributed is initially set up for the DMC version. To patch OF_LINK to run with the original D.P. Johnson version of SDISK3, change offset $0015 from $80 to $00. When running with the D.P. Johnson version of SDISK3, you will find that the system will take a long time and generate lots of retries before reporting disk I/O errors, and when switching disks in a drive from double density to single density, or vice versa, the system will also generate lots of retries before recognizing the new density. This is because only the Sardis Technologies' version of SDISK3 has implemented the "disable retry" bit in the direct sector read/write routines.

* To exit OF_LINK (ie. to exit FLEX or SK*DOS), execute the "MON" command, then OF-MON's "O" command to return to OS-9.

* OF_LINK opens any FLEX virtual volumes as soon as it is invoked, even before the actual booting process begins, so those disks containing virtual volumes should be in the drives before you begin execution of OF_LINK.

* Note that it is possible to specify both a physical and a virtual disk for the same actual floppy disk drive. In that case, the disk containing the virtual volume must be in the drive when OF_LINK begins executing, and when you use "MON"/"O" to exit OF_LINK. At all other times be very careful to ensure that the right disk is in the drive. You should not try to access both the physical and the virtual disk on the same physical drive in a program, as OF_LINK will NOT tell you when to switch disks.

Examples:

```
    of_link /d1@ p=/t2

    of_link /d0@ /d1@

    of_link /d0@ /d2@ /h0/flex/virtual08 p=/p
```

CONTENTS OF THE OF_LINK DISKETTE (OS-9 format)

CMDS contains OS-9 programs:

cmdgen3	Macro command generator
dpoke	Patch a file directly on disk
newcrc	Update header parity and CRC bytes of modules in a file
OF_Link	Multi-purpose program to boot FLEX/SK*DOS, provides device driver linkage, contains OF-MON, etc.

VIRTUAL is a virtual FLEX volume containing these FLEX commands:

BINCPY.CMD	binary file copy
BLIST.CMD	buffered list command
CHECKSUM.CMD	calculate checksum on file
DSKSET.CMD	set disk parameters
DSKSHOW.CMD	display disk parameters
LINK.CMD	link the operating system file for booting
LOADO.CMD	load a binary file with offset
MEMEND.OVR	overlay to change MEMEND to $BBFF (append to FLEX.COR)
P.CMD	print redirection command
PRINT.SYS	dummy printer driver file
PFORMAT.CMD	format a physical disk
VFORMAT.CMD	format a virtual disk
XCOPY.CMD	copies files from FLEX to OS-9, and vice versa

GENERAL INFORMATION FOR ADAPTING SOFTWARE PACKAGES TO RUN UNDER OF_LINK

Programs that use only standard documented FLEX calls should run under OF_LINK without any changes whatsoever. But programs that bypass FLEX and access the hardware directly for such purposes as getting a character from the keyboard without echoing it to the screen, will need modifications.

Some software packages, in their "installation" mode, ask you if you have a standard SS-50 bus configuration (a 6850 ACIA addressed at $E004). If you say "no", they will ask for the address of one or more routines, mentioned below. For example, for input-character-no-echo, respond to the package's prompt with $D3E4.

Other packages require you to write a small I/O driver. The code for input-character-no-echo would be as simple as:

```
        INCHNE JMP  [$D3E5]
```

The following are standard FLEX address vectors:

1) $D3E5-$D3E6 contains the address of a routine that inputs a character from the console without echoing it. Upon return the character is in Register A. OF_LINK also provides a JMP opcode ($7E) at location $D3E4 so the code above could have been written using a direct jump instead of an indirect jump: `INCHNE JMP  $D3E4`
2) $D3F7-$D3F8 contains the address of a routine that checks the console port to see if a character has been keyed on the keyboard. It returns condition code NE if a character has been keyed and is waiting to be read, or EQ if none is waiting.
3) $D3F9-$D3FA contains the address of a routine that outputs a character to the console. Before calling the routine, load the character into Register A.

The guidelines given above should be enough let you install most programs to run under OF_LINK. Some software packages are trickier, and a few of these have more detailed installation instructions given below. If you find other programs, not included below, where the general information given above isn't enough to get the program running, let us know. If you discover the patches needed to get such a program running under OF_LINK, we would appreciate you sending them to us so we can update this manual.

Some of the programs we would like to hear about are: Crunch COBOL, Rapier (chess), PL/9, XDMS, Lucidata Pascal, FORTHBUILDER, FF9 (public domain FORTH from Wilson Frederici), TSC's XBASIC, etc.

ADAPTING THE "DYNACALC" SPREADSHEET TO OF_LINK

During the running of the INSTALL command there are three prompts that need special consideration:

a) DOES YOUR COMPUTER USE S-BUG (OR USE SAME ACIA ADDRESS)?
   Respond with "N".
b) DO YOU -
    1) HAVE TERMINAL ADDRESS
    2) HAVE ADDRESS OF ADDRESS OF TERMINAL
    3) HAVE OWN INPUT CHARAC ROUTINE
    4) QUIT
   Respond with "3".
c) YOUR ADDRESS? D3E4

ADAPTING THE "SUPER-SLEUTH" DISASSEMBLER TO OF_LINK

Change the extension on the CSSINCHR.SWT file to .TXT, then modify it to look like this:

```
* INPUT CHARACTER
INCHRT  STX    BADDR
        JSR    $D3E4
        LDX    BADDR
        RTS
```

Then follow the rest of the manual's instructions to generate an executable program.

ADAPTING THE "SCREDITOR III" WORD PROCESSOR TO OF_LINK

Answer the following prompts in the CONGEN command as indicated:

a) "SYSTEM EXIT VECTOR" - display the contents of $D3F3-$D3F4 and key in that address. Note -- after using Screditor's EX command, use OF-MON's "F U hhhh" command to return to Screditor.
b) "SCREEN TYPE" - use "2" for external character display routine
c) "SCREEN DRIVERS COMPATIBLE" - answer with "Y"
d) "ADDRESS OF SCREEN OUTPUT" - display the contents of $D3F9-$D3FA and key in that address.
e) "ADDRESS OF SCREEN INIT" - display the contents of $D3F5-$D3F6 and key in that address.

Answer the following prompts in the KEYGEN command as indicated:

a) "KEYBOARD TYPE" - use "5" for external routine using NE
b) "META KEYS" - answer with "N"
c) "ADDRESS OF KEYBOARD CHECK" - display the contents of $D3F7-$D3F8 and key in that address.
d) "ADDRESS OF GET CHARACTER" - key in address "$D3E4"
e) "MUST ECHO BE CONTROLLED" - answer with "N"

OF_LINK

f) "KEYBOARD DELAY CONSTANT" - experiment with values between 10 and 30.

Answer the following prompts in the PRTGEN command as indicated:
a) "PRINTER OUTPUT ADDRESS" - respond with "$CCE4"
b) "PRINTER INIT ADDRESS" - respond with "$CCC0"
c) "PRINTER LEFT MARGIN" - respond with 000 or whatever value you require
d) "NULLS AFTER CR-LF (NEW LINE)" - respond with 000 (use OS-9's XMODE
   instead to change the "null" value of the printer descriptor, if
   necessary)

ADAPTING THE "STYLOGRAPH" WORD PROCESSOR TO OF_LINK

Carefully study the section towards the end of the Stylograph manual entitled "Using Stylograph with the FLEX disk operating system". Follow its instructions on modifying the STYIO file. Use the following code as a guide:

```
             ORG    IOBEG

     *CONSTANTS
     RAMLIM  FDB    $FFFF     HIGHEST RAM USED IF LOWER THAN MEMEND
     SIMFLG  FCB    1         NON-ZERO IF NO INTERRUPTS
     PRTFLG  FCB    1         ZERO IF OUTPUT THROUGH FLEX, ELSE PRTOUT
     INTVEC  FDB    0         INTERRUPT VECTOR LOCATION

     INCHEK  EQU    $D3F7     FLEX - check status of keyboard port
     INCHN   EQU    $D3E5     FLEX - input character from keyboard, no echo
     OUTCH   EQU    $D3F9     FLEX - output character to console screen

             ORG    IOBEG+$10 SET BRANCH TABLE
     *BRANCH TABLE
     INTON   BRA    JINTON
     INTOFF  BRA    JINTOFF
     PINIT   BRA    JPINIT
     PCHECK  BRA    JPCHECK
     GETCH   BRA    JGETCH
     PRTOUT  EQU    *

     *OUTPUT CHARACTER
             JMP    [OUTCH]

     *GET A CHARACTER
     JGETCH  JMP    [INCHN]

     *CHECK FOR CHARACTER AT INPUT
     JPCHECK JMP    [INCHEK]

     * ENABLE INTERRUPTS
     JINTON  RTS

     * DISABLE INTERRUPTS
     JINTOFF EQU    JINTON
```

```
*INITIALIZE PORT
JPINIT  PSHS   A          flush FIFO
JP10    JSR    [INCHEK]
        BEQ    JP90
        JSR    [INCHN]
        BRA    JP10
JP90    PULS   A,PC
```

ADAPTING THE "RMS" DATA MANAGEMENT PACKAGE TO OF_LINK

1) Load RMS.CMD into memory with the FLEX "GET" command, then use the "MON" command to exit FLEX and enter OF-MON.
2) Look at the following locations and verify that they contain the values in the "OLD" column:

address	old	new	description
0022	E0	don't care	address of console port MSB
0023	04	" "	" " " " LSB
1B87	34	6E	routine to input character from console port
1B88	10	9F	
1B89	9E	D3	
1B8A	22	E5	
1B94	35	don't care	
1B95	10	" "	
1B96	39	" "	
1B97	34	6E	routine to output character to console port
1B98	14	9F	
1B99	9E	D3	
1B9A	22	F9	
1BA3	35	don't care	
1BA4	14	" "	
1BA5	39	" "	

3) If the "old" values agree, patch RMS with the "new" values at the locations indicated.
4) If the "old" values don't agree, you have a different version of RMS than the one we tested. Use OF-MON's "S" command to search for the same code residing in a different location.

FF9 (FORTH-83 from Wilson Frederici)

1) If you want to use a FLEX virtual disk for a FORTH disk, you will need to change 3 of the 4 constants involved in the conversion of disk block numbers to sector locations:

```
     TRK/DRV = 1 to 127      (was 35)
     SEC/TRK = 64            (was 10)
     BASETRK = 1             (was 0)
```

   The TRK/DRV (tracks per drive) value depends on the size of the virtual volume.

INTRODUCTION TO THE OF-MON MONITOR PROGRAM

OF-MON is entered by exitting FLEX or SK*DOS with the "MON" command.

OF-MON, with its lack of breakpoints and register commands, is not intended to be a full fledged debugging monitor -- rather, its main purpose is to allow contents of memory to be examined and changed. If you need powerful debugging capabilities, we highly recommend TSC's excellent "6809 DEBUG" package available from some of the suppliers listed elsewhere in this manual.

OF-MON Monitor Commands

Commands consist of a one character code followed by 0 - 3 arguments. Some arguments are one or two alphabetic characters, but most are entered in hexadecimal, either 2 or 4 digits each. Keying any non-hex digit (such as a carriage return) when a hexadecimal value is required, aborts the command. Here is a summary of the available commands:

M	Memory examine/change byte by byte
F	return to FLEX
O	return to OS-9
J	Jump to subroutine or program *
E	Examine large blocks of memory *
S	Search for byte *
B	move Block of memory *
A	fill Area of memory *
?	display menu of OF-MON commands

(* not implemented yet)

"A <begin address> <end address> <value>" -- Fill Area of Memory

The memory fill command allows the user to fill a contiguous area of memory, from the begin address to the end address, inclusive, with the 1 byte data value entered. Note -- be careful not to overwrite FLEX or OF_LINK; the destination area should normally avoid $C000 - $FFFF.

Example:
```
=A 2000 7FFF 3F
```

"B <source-begin address> <source-end address> <destination-begin address>" -- Move Block of Memory (ie. duplicate it)

The move block command copies the contents of the contiguous area of memory, source-begin to source-end addresses, inclusive, to a same-sized area starting at the destination-begin address. If the source and destination blocks overlap, the results might not be what was desired. If no overlapping occurs, the contents in the source area remain unchanged. Note -- be careful not to overwrite FLEX or OF_LINK; the destination area should normally avoid $C000 - $FFFF.

Example:
```
=B DE00 DFFF 1000
```

"E <begin address> <end address>" -- Examine Block of Memory

The examine memory command displays the contents of memory from the begin to the end address, inclusive. The contents are displayed in both hexadecimal and ASCII. Each line shows 16 bytes and is aligned on an even 16 byte boundary (ie. the first displayed byte of each line has an address ending with "0"). Non-printable characters ($00-$1F, $7F) are shown as dots in the ASCII display. A sample display looks like this:

```
=E FE7F FE96
FE70 250B8146 22078037 39326132 61433953 %..F"..792a2aC9S
FE80 542D4D4F 4E20312E 30202863 29203139 T-MON 1.0 (c) 19
FE90 38342044 4357043D 043F3F04 0D0A0000 84 DCW.=.??.....
```

The display can be temporarily halted at any time by pressing the "Escape" key on the keyboard (control-BREAK on the CoCo 3). At this point, pressing the "Return" key will abort the command; typing any other character resumes the display.

"F W" -- Return to FLEX Warmstart

The "F W" command is used to return control to FLEX (or SK*DOS) via its Warmstart address ($CD03) after having exitted FLEX by using the "MON" command.

"F U <address>" - Return to FLEX environment, but to user program

The "F U hhhh" command is used to jump to a user program running in the FLEX (or SK*DOS) environment. That is, the program uses FLEX routines and/or will eventually return control to FLEX's "Warmstart" address ($CD03).

A typical use for this command is where you want to modify a few bytes of a program before executing it. Use FLEX's "GET" command to load the program from disk, use "MON" to exit to OF-MON, use the "M" command to make the desired patches, then use the "F U" command to jump to the entry point of the program. If the documentation of the program does not tell you the location of its entry point, you will first have to run TSC's "MAP" command to find out.

Example:
```
=F U C100
```

"J <address>" -- Jump to subroutine or program

The jump to subroutine (or program) command lets the user execute a routine beginning at the specified address.
Return to OF-MON should be via an "RTS" instruction at the end of the subroutine or program, not a jump.

Note -- do NOT use this to return to FLEX (or SK*DOS), or to execute a program that will use FLEX routines or will return to FLEX warmstart ($CD03). Use the "F W" or "F U" commands instead.

Example:
```
=J 0100
```

"M <address>" -- Memory Examine/Change

The memory examine/change command lets the user look at and modify the contents of memory, one byte at a time, beginning with the specified address. Each byte is shown on a separate line, with its address, its current contents in hexadecimal, and its current contents in ASCII (if printable). At this point you can key one of several sub-commands:

a) typing a valid 2 digit hexadecimal value changes that memory location to the new value (a "?" is displayed if not successfully changed because of bad RAM or because it was ROM, or it was changed OK but cannot be read back because it was a write-only register in an I/O chip), then advances to display the next location. Note -- if the address was in the $FF00-$FFFF range, the command is exitted after the one location was changed.

b) typing an apostrophe followed by one ASCII character stores that ASCII character into the memory location.

c) typing a comma or space advances to the next location without changing the current location

d) typing a backspace or minus sign or up arrow displays the previous location instead of the next one -- the current location is not changed

e) typing a period or carriage return exits the command -- the current location is not changed

f) if you type anything else you will get a question mark, and the current location will be redisplayed.

Entire machine language programs can be entered this way. But be especially careful when changing any bytes in areas occupied by FLEX or OF_LINK or OS-9 ($C000 - $FFFF).

"O" -- return to OS-9

Exit OF-MON to return to OS-9. This will terminate the current session of FLEX/SK*DOS. If any FLEX logical drives are using virtual disks, the corresponding OS-9 files will be closed. Using OF-MON's "O" command is the "proper" way to return to OS-9 from FLEX.

"S <begin address> <end address> <search value>" -- Search for byte

The search command finds all occurrences of the specified 1 byte data pattern between the specified begin and end addresses, inclusive. Each "hit" is displayed on a separate line, with its address, the two bytes before the search value, the search value, and the two bytes after. Note that the address is that of the middle byte. A sample display looks like this:

```
=S C100 C12F BD
C114 0000 BD CD24
C11A C840 BD CD2D
```

The display can be temporarily halted at any time by pressing the "Escape" key on the keyboard (control-BREAK on the CoCo 3). At this point, pressing the "Return" key will abort the command; typing any other character resumes the display.

SOURCES OF SOFTWARE FOR FLEX AND SK*DOS

- Southeast Media / '68' Micro Journal
  5900 Cassandra Smith Rd.
  Hixson, Tennessee 37343
  U.S.A.
  (615) 842-4600

  Southeast Media recently claimed to be the largest 68XX software distributor in the world, with over 300 programs available for a wide variety of systems. Current issues of "'68' Micro Journal" magazine, published by a different division of the same company (and same address as Southeast Media) contain a 7 page mini-catalogue of software, much of it available for FLEX. Besides their software sales, this is one magazine you may wish to subscribe to, as it is probably the only one left that even mentions FLEX or SK*DOS. Several dozen disks of user submitted programs that have been published in previous issues of '68' Micro Journal are also available for $16.95 ($US) each -- a complete listing appears in each issue.

- Frank Hogg Laboratory, Inc.
  770 James Street
  Syracuse, NY 13203
  U.S.A.
  (315) 474-7856

  Frank Hogg Labs have also claimed at times to be the largest 68XX(x) software distributor. They used to support FLEX extensively, and their "Serious Users Software Catalog" was chock full of assemblers, compilers, disassemblers, debuggers, editors, word processors, spelling checkers, data base managers, spread sheets, accounting packages, etc. Their emphasis has now shifted to OS-9, but you may want to still send for their free newsletter and catalog.

- AAA Chicago Computer Center
  120 Chestnut Lane
  Wheeling, Illinois 60090
  U.S.A.
  (312) 459-0450 (until Nov. '88)
  (708) 459-0450 (after Nov. '88)

  AAA is also mostly a dealer for other companies' products, both hardware and software, but have in the past also offered a few of their own software packages. Ask them for a catalog.

- Star-K Software Systems Corp.
  Box 209
  Mt. Kisco, NY 10549
  U.S.A.
  (914) 241-0287

  This company wrote SK*DOS (a FLEX "clone"), and also offers some other software to run under FLEX or SK*DOS.

- Computer Systems Consultants, Inc.
  1454 Latta Lane
  Conyers, GA 30207
  U.S.A.
  (404) 483-4570 or 1717

  Various programs, such as the Super-Sleuth disassembler, cross-assemblers to generate code for many other microprocessors, debugging simulators, assembler code translators, modem telecommunications program, etc. See their ad in '68' Micro Journal, or send for a catalog.

- Wilson Frederici
  1208 NW Grant
  Corvallis, OR 97330
  U.S.A.

  An excellent public domain FORTH-83 interpreter called FF9 is available from Wilson for next to free -- just send him two blank 5.25" disks and a prepaid mailer!!

- Other suppliers of FLEX software also advertise in '68' Micro Journal.

- Another often overlooked potential source of inexpensive FLEX software is those people who used to run FLEX but have since switched to other operating systems and haven't found time to run FLEX any more. There must be a lot of good used software packages gathering dust in peoples' closets. Try the classified ad section in magazines such as '68' Micro Journal and MOTD (the OS-9 User Group newsletter), or leave messages on computer bulletin board systems.

SYSTEM MEMORY MAP

```
0000 - BFFF  * User memory
C000 - D36F  * Various components of FLEX ($C71C-$C83F not used)
D370 - D3E1    (not used)
D3E2 - D3E4    console and timer I/O values
D3E4 - D3FC  * Console and timer address vectors
D3FD - DDFF  * Various components of FLEX
DE00 - DE1D  * Disk driver jump table
DE1E - DF51    Disk driver routines
DF52 - DFFF    (not used)
E000 - FDFF    OF_Link (an OS-9 module)
FE00 - FEFF    OS-9 vector page
FF00 - FFFF    I/O, GIME registers, etc.

             * These areas are standard FLEX -- refer to the FLEX User's
               Manual and FLEX Programmer's Manual for more details.
```

....