Win95 FLEX Emulation System

Written by Ben Williams, ars:AA7AS, ex-N4EJI

EMAIL: bwilliams@blackbelt.com
WEB: http://www.blackbelt.com/blackbelt/flexem.html
FTP: ftp.blackbelt.com/corporate/blackbelt/

Revision: 1.33 - May 31st, 1996
Executable Type: Windows 95 Native Application

Many, many Thanks To: Pete Gerardi - ars:KE4TP, ex-WB4YQK
Tim Heffield - ars:N4IFP
Bob Phillips @ GIMIX Corporation, Chicago
Frank Hogg @ Frank Hogg Laboratories

Overview

What is this?

The package is a complete emulation of a 6809 processor based system, running the TSC Flex operating system over the Percom PSYMON monitor "ROM" with a useful command extension set. It runs under Win95. On a modern PC (Pentium, 60 mhz or better) the emulator is faster than the original computer system was.

The 6809 is a 64k memory capable microprocessor with a decent register set and a very strong instruction set for an 8-bit microprocessor of its day. Motorola was the source of the design of the 6809; it was derived, somewhat loosely, from the 6800 and offered source code compatibility for the most part. The next step beyond the 6809 was the 68000 series.

Flex is a small (4k!) disk operating system (DOS) which, along with CP/M and a few others, truly pioneered small "personal" computers. These were the systems that were in place when Apple Computer started in a garage.

PSYMON is a "system monitor" that nominally is 1k. In this emulator, there is an additional 1k ROM extension which I created. It provides some additional capabilities to the monitor.

By "complete", I mean that I've provided...

- A very comprehensive emulation of the original computer system.
- All the commands and utilities you'd need to do real work.
- Extremely high accuracy of emulation.
- Usable speed

If you ran Flex in the past, you'll likely experience a strong sense of nostalgia within seconds after starting the software. Welcome to the club!

For the future, I hope to provide more Flex commands and other useful items to enhance the overall usability of the package. I have a large number of diskettes I cannot read at this time due to a lack of the correct diskette drives for my "physical" Flex system. I am trying to obtain 80 track drives; If and when I do, I have a *lot* more that can be included in the emulation release.

Feature List

— Extremely stable and reliable
— Includes useful set of Flex commands and utilities
— Supports up to four virtual disks, fully cached
— Virtual disk sizes from 85k to 16 megabytes supported
— Complete set of Host-based virtual disk maintenance utilities
— Printer support via Flex "P" command via Host printer
— Flex system time, date set from Host clock on startup & once/minute
— 2 editors, BASIC, assembler, disassembler, utilities, much more
— Host-based startup script automates logging virtual disks on
— Host environment is custom terminal look-alike - deja vu!
— Abortable so system problems cannot harm virtual disks
— "Quit" and "Abort" Flex commands for exit from keyboard to Host
— "Flush" Flex command to write out virtual disks immediately
— 6809 "RESET" menu command to recover control if you have an "oops".
— Well documented (assuming you know how to use Flex, that is)
— Memory efficient operation (but watch those disk sizes!)
— Full-boat 63k RAM 6809 virtual machine, plus 2k virtual ROM
— ACIA Serial port emulation @ $E004/$E005 & "mirror" @ $E008/$E009
— Complete 6809 instruction set emulation (except "HALT")
— Emulates SOROC IQ-140 terminal (so screen editor can work)
— Runs at about 3.4 mhz 6809 equivalent on a 133 mhz Micron Pentium
— Traps on ALL illegal opcodes (they convert to SWI, $3F).
— PSYMON monitor runs on "Mon" command or menu RESET command
— PSYMON source code included on virtual disk
— Completely isolated from host, totally safe for experimenting

Distribution

You'll find these files in the emulator archive:

flex.exe -	The emulator itself
flex.bin -	6809 Flex DOS, PSYMON low level machine code monitor
mad.exe -	"Make A Disk" virtual disk creator & formatter in Win95 shell
fdir.exe -	Displays directories of Flex virtual disks in Win95 shell
inject.exe -	Transfers files Host->Flex, in Win95 shell
distill.exe -	Transfers files Flex->Host, in Win95 shell
disk0 -	Virtual system disk for Flex
disk1 -	Virtual work disk for Flex
startup.txt -	Flex startup commands (NOTE: resides under Host OS!)
readme.doc -	This file
readme.txt -	Text file suitable for reading in an editor

This software provides emulation of all 6809 official* instructions with the exception of "HALT". The virtual computer which the emulator creates has 64k of memory, 2k of which is "ROM", or read-only. There is also a low-level "monitor"; this is called "PSYMON". It originally came from Percom corporation.

* There are some undocumented instructions that work, or partially work, within some versions of the 6809 chips. These are NOT supported in the emulator; Motorola, the architects of the 6809, never supported or admitted to these various operations. Software that depends on these features will not function as intended. Also, Hitachi created a chip called the "6309" which had some additional functionality. This is not emulated at this time.

The software provides character I/O via emulation of a Motorola 6850 ACIA chip located at addresses $E008 and $E009, and aliased at $E004 & $E005. It provides Disk I/O via emulation of a powerful DMA

disk controller that lets the Host do 99% of the work using the Host processor directly, which means that disk I/O is considerably faster under the emulation than it was for any conceivable 6809 system. Unfortunately, this is pretty well hidden by the enormous amount of work that the emulation itself must do unless you have a reasonably modern processor (Pentium @ 60 Mhz or better). Printer support is likewise handled by the Host processor directly as a DMA operation. Under Flex, you just use the "P" command as you usually would.

Where are they now?

TSC is out of business, its principals scattered, and the source code for the operating system appears to be lost. I do have a lead on it, but thus far it has not panned out. I was also unable to locate Percom; perhaps they too have gone out of business - but I've not been able to confirm this. The number of Flex systems in use today is certainly a very small number. Other companies that were involved in Flex-based computers are still around (such as GIMIX & Frank Hogg Laboratories) but they understandably have moved on to other projects. I contacted both companies I just mentioned, and while the principals were very helpful, no commercial activity takes place with regard to Flex any longer.

So why do this?

There are several reasons.

First, I personally have a great deal of interesting 6809 source code around from the early 1980's when I was designing 6809 based arcade games, and I wanted to be able to experiment with that code. I have some of the actual game machines also, so by experimenting I mean making changes and playing about with the actual games themselves.

Second, although I was able to get an SS-50 based machine up and running Flex, I had a serious concern that if the hardware failed, I might be unable to get it running again due to a lack of available parts. With this emulation, Flex has a new lease on life.

Third, the emulation itself makes available a very powerful "small" computer system inside the Host which is actually a very nice, very safe environment for playing around with assembly language programming. You can't hurt the Host at all no matter how bad a mistake you make within the emulation, and as a computer, the 6809 sports extremely powerful addressing modes, a decent register set, and a solid instruction set which makes experimenting with it very enjoyable. Since the disks are cached in the Host's memory, even if you foul up the disks, you can still abort the emulator and you're 100% safe - the disk information will be intact.

Finally, it was a challenge to write, and then to port, the emulation and get it all working. Overall, it's been a task which provided me with a great deal of personal satisfaction.

The results - the complete emulation - appear to me to clearly have absolutely no commercial value, and although the companies that produced the software which run within the emulation are long gone, I am extremely uncomfortable with the idea of selling something that contains, and depends upon, commercial work that they did long ago. The alternative is to have Flex fade away into the past, forgotten, which would be a shame - Flex was an important part of the computer revolution. For these reasons the entire project is freeware. I hope that this project will bring some enjoyment to someone, somewhere. It certainly has done so for me!

How it works

Startup

1 - After extraction, copy all files to a subdirectory, or extract them there in the first place. I suggest you make a directory for this application... there are enough files in here that it's probably best if you don't mix them up with other applications.

2 - Use Win95 to add the program "FLEX.EXE" to the Start button or to a drawer, depending on how you prefer to use Win95. The Flex emulator has its own distictive icon; it also has a small "+++" icon which is visible in the title bar and when it is minimized. When Win95 starts the emulator, either from a shortcut or from the "real" icon, it passes the directory where the emulator is located *to* the emulator, and that is where it will look for the virtual disk files and the startup.txt file.

When the emulation starts, PSYMON and Flex are loaded into emulator memory from the current directory and initialized. An external text startup file is used (startup.txt, from the same directory as the emulator and its other files) to supply an initial set of commands to Flex in order to get things the rest of the way up and running. You'll see the following when the emulator starts:

```
+++LOGIN DISK0 0                    (new built-in command)
+++LOGIN DISK1 1                    (new built-in command)
+++DATE                             (command from DISK0 )
<current date is printed>
```

These commands are coming from the "startup.txt" file which is a Host OS file. Do not change this file until you completely understand what is going on with the LOGIN commands. These are new to Flex! See the next section for details.

Virtual Disks & Virtual Drives

Because the emulation uses virtual disks (these are the "disk0" and "disk1" files) you need a way inside Flex to "change" disks. There are two new built in commands within this version of Flex to do this: "LOGIN" and "LOGOUT". To put a disk in a virtual drive, you use "LOGIN". To remove a disk from a virtual drive you use "LOGOUT". The syntax for these commands is as follows:

```
LOGIN <HostDiskFileName> <FlexDriveNumber>
LOGOUT <FlexDriveNumber>
```

When logging in a new disk, any disk that was previously in that drive is logged out automatically.

To save current changes to the virtual disks without exiting the emulator, execute the FLUSH command or the Flush menu selection.

When you exit the emulator via the QUIT command or Exit menu selection, all virtual disks are logged out automatically. When you exit the emulator via the ABORT command or the Abort menu selection, the changes you have made to the virtual disks during this emulator session, or since the last use of the FLUSH command or menu selection, are lost.

Creating Virtual Disks

You may notice that there is no "FORMAT" or "NEWDISK" command on the Flex disks. This is because the disks are created outside of Flex, under the Host OS. This is natural because the Host OS is the home of the virtual disk files, not Flex. The Disk formatting utility command MAD (Make A Disk) is provided and has the following parameters:

```
<required> [optional]
---------------------
 MAD <HostFileName> <FlexDiskName> <FlexDiskNumber> <Tracks> <Sectors>
```

HostFileName is the name that will be given to the virtual disk under the Host OS. FlexDiskName is the name that will be given to the virtual disk under Flex. FlexDiskNumber is the volume number under Flex. Tracks is the number of tracks that the disk will have; sectors is the number of sectors per track the disk will have. These two parameters together determine the size of the disk in the following manner:

```
(TRACKS - 1) x SECTORS x 252 = User Data Bytes On Disk
```

Standard disk sizes under Flex were 35, 40 or 80 tracks, with 10, 20 or 40 sectors per track. This table shows you how much room these standard sizes provided:

Tracks	Sectors/track	Sectors/disk	Bytes	Overall Drive Configuration
35	10	340	85,680	35 track, Single Sided, Single Density
35	20	680	171,360	35 track, Single Sided, Double Density
35	20	680	171,360	35 track, Double Sided, Single Density
35	40	1360	342,720	35 track, Double Sided, Double Density
40	10	390	98,280	40 track, Single Sided, Single Density
40	20	780	196,560	40 track, Single Sided, Double Density
40	20	780	196,560	40 track, Double Sided, Single Density
40	40	1560	393,120	40 track, Double Sided, Double Density
80	10	790	199,080	80 track, Single Sided, Single Density
80	20	1580	398,160	80 track, Single Sided, Double Density
80	20	1580	398,160	80 track, Double Sided, Single Density
80	40	3160	796,320	80 track, Double Sided, Double Density

If you're paying attention, you may have noticed that the calculations shown here are "short" by one track.

The track that is not counted in these calculations is used in all formats to hold the directory of files for the disk. In unusual cases, there may be enough files on a disk to require more directory space than the first reserved track; in that case, Flex will automatically add sectors to the directory from the currently unused data sectors. Alternatively, there is a utility provided called "extend" which will reserve a block of sectors if you know ahead of time that you will be placing an unusually large number of files on a disk. Extending a directory this way makes the directory searches run faster on a floppy system; within this emulated system, the file I/o is not noticeably affected by fragmentation.

"Normal" Disks

The MAD utility can create disks with any reasonable format; and the emulator's disk drivers for Flex can read any disk MAD is willing to create regardless of the format. For the reason that more simplistic Flex utilities may expect certain sizes, unless you have a particular need for an unusual size, we suggest that you stick with one of the standard sizes from the table above.

Large Virtual Disks

Using the MAD utility, you can create disks with up to 255 tracks and 255 sectors. This is about 16 megabytes. Remember that the disk files actually take up that much space (unless the Host OS has disk compression) - so don't do this just for fun. But if you're doing something that needs lots of disk space, you can meet the challenge.

Multiple Virtual Disks

Flex was pretty much designed for systems with a minimum of two floppy drives. There are no well thought out facilities for copying files using only one drive, for example. That's why the emulator is configured to start with two drives. If you like, you can add up to two more drives (drives 2 and 3) using the LOGIN command. Drive sizes may be mixed and matched in any way, keeping in mind of course that when you're copying from one disk to another, the target disk needs to have enough room to hold everything you're copying. Also, each disk that you log in takes up as much of your ram as it has bytes; so unless you have scads of memory (hey, I do... doesn't everyone?) you'll want to be thoughtful about how many disk you keep on line at once.

Exiting the Emulator and Backing up the VDisks

Exit Gracefully:

At the Flex command prompt, type "QUIT" (without the quotes) and press return; or select Exit from the File menu; or press the close button at the top right of the emulator window; or select Close from the System menu. Any of these actions result in a graceful exit.

Exit, Not so Gracefully:

At the Flex command prompt, type "ABORT" (without the quotes) and press return; or select Abort from the File menu. Either of these actions will exit the emulator and discard any changes made during this session, or since the last use of the FLUSH command or menu selection.

Backup the contents of the VDisks without exiting:

Type FLUSH at the Flex command line; or select Flush from the File menu. This will rewrite the virtual disks back to the Host OS immediately, if the disks have been changed. This is a very good thing to do when you're about to test something you don't have complete faith in.

Multitasking Issues

Because of the way that Flex was written, there is no effective way to put the emulator to "sleep" while it is waiting for a character and guarantee that this is not debilitating a process.

Several issues prevent this; for instance, when the screen editor runs, it constantly checks the ACIA for input while redrawing the display. Putting the task to "sleep" until a character is input would stop the editor from functioning.

The Win95 version of the emulator does obey the Win95 "bible" for multitasking as a busy process; that means other software will share time equally with the emulator.

There is an option in the Execute menu called Suspend which tells the emulator to stop execution until the next time Win95 gives it a time slice, each time it checks to see if a character is ready, unless there actually is a character ready, in which case the emulator continues to process. This appears to be the best compromise between full-time execution and "sleep on wait for input" for the emulator. Using the Suspend option will not, in most cases, significantly affect the emulator. The default is on - the emulator will give up the thread when checking for a character and no character is input.

The way the Suspend mechanism is built ensures that each time Win95 gives the emulator a time slice, enough 6809 code will run to re-check for a character, ensuring immediate an response to the user.

Printer Support

Flex supports printing via the "P" command prefix. For instance, if you type:

```
LIST FILENAME
```

The file is listed to the monitor. If you type:

```
P LIST FILENAME
```

Then the file is sent to the printer. This version of Flex drives the printer (via LPT1: and the emulation software).

> ***Postscript users please note:***
> Because of the limitations of the LPT1: output model provided by Microsoft, Postscript printers will NOT work with this emulation - the output to LPT1 goes DIRECTLY to the printer, which proceeds to ignore it or misunderstand it completely. You need a character-mode printer such as an HP Laser or compatible in non-PS mode.

Obtaining Listings of Files inside Virtual Flex Disks

When you are working under the Host OS's command shell, the command "FDIR" will provide you with a listing of the files contained in a virtual Flex disk. Syntax is as follows:

```
<required> [optional]
---------------------
FDIR <DiskName>
```

Copying files into Flex Virtual Disks from the Host OS

We have supplied the utility command "INJECT". INJECT has the following parameters:

```
<required> [optional]
---------------------
INJECT <SourceName> <DiskName> <DestName> [T]
```

SourceName is the filename of the file to be injected under Host OS, DiskName is the name of the Virtual Disk file under the Host OS, and DestName is the filename of the file to be created in the virtual disk under Flex. The optional T parameter deals with text translation, discussed next.

Note that text files under Flex use the Carriage Return character as an end of line marker. If you supply the "T" parameter to the INJECT command, Line Feeds in incoming files are discarded as the file is transferred to the virtual Flex disk. Otherwise, you will have to change the characters yourself. If you're transferring a binary file, you should never use the T parameter, or the file will be corrupted.

The INJECT command will not allow you to copy a file into a Flex virtual disk if that filename already exists within the directory of the virtual disk. You have two choices of remedy when this occurs: use INJECT with a different target filename, or execute the emulator and rename or delete the previously existing file.

Obtaining files from a "real" Flex System

This is difficult because it requires some kind of command over on the Flex system itself.

We provide a Flex command called "PUSH" (which you will find on the disk0 virtual disk as PUSH.CMD and on the disk1 virtual disk as PUSH.TXT, which is the 6809 assembly source code). PUSH takes any Flex file and sends it to the console terminal as a series of well formatted HEX numbers. The correct procedure is to:

1 - Use the Host as a serial terminal on the Flex system;

2 - "Capture" the output of the PUSH command as it's filename;

3 - Edit the Capture file to remove any extraneous information such as the +++ prompt, unrelated commands, blank lines, and so on;

4 - Turn the captured data back into binary. Somehow.

The problem here is that the PUSH.CMD file must exist over on the flex system. So you will need to copy the PUSH.TXT file into your Flex machine and assemble it in order to transfer files. That's fairly easy to do as long as you have a 6809 assembler over there... it's short and sweet.

Copying files from Flex Virtual Disks to the Host OS

We have supplied the utility command "DISTILL". DISTILL has the following parameters:

```
<required> [optional]
---------------------
DISTILL <SourceName> <DiskName> <DestName> [T]
```

Where SourceName is the filename of the file to be distilled under Flex, DiskName is the name of the Virtual Disk file under the Host OS, and DestName is the filename of the file to be created in the Host filesystem. The optional "T" parameter deals with text translation, as described above for the INJECT command.

> ***Note:*** Flex compresses text files in a special way; the TAB character (0x09) is used as a signal that the next byte in the file is a count of compressed spaces. So, 0x09,0x04 means four spaces - not the ASCII sequence, "TAB", "EOT". DISTILL will automatically decompress files with a .TXT extension. The only time this could cause problems is if a binary file had a .TXT extension, which you should of course avoid. This would also cause problems within Flex. Text files also can contain NULL characters at the end; the decompressor in DISTILL will automatically remove these if you use the T option. Otherwise, you have to deal with the space compression and the final nulls.

Questions or Ideas?

Please write to me if you feel have something to contribute to this project.

Here's how to contact me:

By Post:

Ben Williams
Black Belt Systems, Inc.
398 Johnson Road
Glasgow, MT
59230

By email:

bwilliams@blackbelt.com

Bugs

Bugs? There are no bugs in this software, whatever can you be thinking? Seriously, if you find anything (like an unusual 6809 instruction that isn't emulated properly), let me know and I'll fix it ASAP. I have a very strong interest in seeing the emulation run flawlessly.

Appendixes

Appendix A - Terminal I/O

The serial terminal output emulation simply uses a minimal emulation of a SOROC IQ-140 terminal. The SOROC emulation is provided specifically so that the STYLUS editor will work. Another change made to the emulator for STYLUS was the address "mirroring" of the ACIA to address $E004 & $E005, since STYLUS hits the ACIA directly in order to obtain max speed; although this is very bad coding practice, it was more or less understandable in the day this was written.

Here are the controls available in the terminal emulation:

Char Sequence	Operation or Effect
$07	Bell
$08	Backspace
$0A	Line Feed
$0D	Carriage Return
$1E	Home Up
$1B,E	Insert Line
$1B,R	Delete Line
$1B,T	Erase EOL
$1B,t	Erase EOL
$1B,*	Home Up & Erase EOF
$1B,Y	Erase EOF
$1B,y	Erase EOF
$1B,(	Set Write MAGENTA ("low" intensity)
$1B,)	Set Write GREEN ("high" intensity)
$1B,=,yy,xx	Write Cursor Position ("xx" & "yy" are 0x20 + cursorvalue)

Appendix B - Documentation for Flex DOS

Here are some very basic docs for the Flex commands:

```
<> = REQUIRED    [] = OPTIONAL

H = HEXADECIMAL NUMBER DIGIT
D = DECIMAL NUMBER DIGIT
T = ASCII TEXTFILESPEC = [drivenumber]<filename>[extension]

examples:  0.COPY.CMD
           COPY
           COPY.CMD
           0.COPY

DRIVESPEC = [DRIVENUMBER]
examples:   0
            1
            2

MATCHLIST = [PARTIAL FILENAME][EXTENSION]
examples:   C
            C.CMD
            CO
            CO.CMD
            COPY.CMD

DRIVELIST = <DRIVENUMBER> [DRIVENUMBER] [DRIVENUMBER]...
examples:   0
            0 2
            0 1 3
            0 1 2 3
```

Flex Command List

```
   ABORT (aborts emulator - LOSES all changes to virtual disks)
  APPEND <FILESPEC> <FILELIST> <FILESPEC>
    ASMB <FILESPEC> [FILESPEC] [options] (you need to know 6809 Assembly
Language)
     ASN [W=DRIVENUMBER] [S=DRIVENUMBER]
   BASIC (complete basic interpreter, with external online help via HELP
command)
   BUILD <FILESPEC>
     CAT <DRIVELIST> <MATCHLIST>
   CHECK <DRIVE>
    COPY <FILESPEC> <FILESPEC> or COPY <FILESPEC> <DRIVE> or COPY
<DRIVE> <DRIVE> [MATCHLIST]
   DASMB (help is available inside the disassembler... press ?)
    DATE [MM-DD-YY]
   DEBUG (no help available... good luck - I can't remember how this
works!)
  DELETE <FILESPEC> [FILELIST]
     DIR [+P] [DRIVELIST] [MATCHLIST]
  DISKEX (no parameters... commands are:
            ^ - Scroll forward through current sector
            A - New track & sector to process
```

```
            C [HH]- Change data in sector
            D - New disk to process
            F - Forward link to next sector (using link in sector)
            N - Next Sector
            P - Previous Sector
            S - Scroll Value
            W - Write sector back to disk (BE CAREFUL!!!!)
            X - Exit to Flex)
    DUMP <FILESPEC>
    ECHO <TEXTSTRING>
    EDIT [FILENAME] (fully WYSIWYG text editor... an amazing tool)
              - Menu driven; press control-A for help in edit mode.
      EX <FILESPEC>
  EXTEND <DRIVESPEC> <SECTORS>
   FCASE (flips state of upper case control, which defaults to on)
   FILES <DRIVESPEC> [MATCHLIST]
FILESORT (file cataloging utility by Brian Bailey)
   FLUSH (Write VDisks out to Host OS NOW)
    FREE <DRIVESPEC>
 FREEMAP <DRIVESPEC>
   HECHO <HH> [HH] [HH] [HH]...
    HELP [COMMAND] (online help system uses .DIR files)
       I <FILESPEC> <flex command...> (takes input from file, not
console)
    JUMP <HHHH>

   LEDIT <FILESPEC> (line based text editor... basic commands include:
        [DD]       - New current line number
            B      - Goto Bottom of file
        [DD]C"T"T" - change text in line (any delimiter works)
        [DD]D[DD]  - Delete line(s)
            H      - Display Horizontal Ruler
        [DD]I[DD]  - Insert at line DD (#<return> to exit insert mode)
        [DD]O      - Overlay text on current line
        [DD]P[DD]  - Display from current line, DD lines
        [DD]P!     - Display from current line to end
        [DD]P[dd]  - Display from DD, dd lines
        [DD]R      - Replace Line
            REN    - Renumber all lines
            S      - Save file and exit
            T      - Goto Top of file
        [DD]=[T]   - current line becomes replacement text

    LIST <FILESPEC>
     MAP <FILESPEC>
   MEMEX (no parameters... commands are:
            A <HHHH> - New address
            C <HHHH> - Change data at address
            S <HH> - Set new scroll value
            X - Exit to Flex)
       N <flex command> (answers all questions with N)
    NAME <DISKNAME>
       O <flex command> (directs output to Flex file)
       P <flex command> (directs output to Host printer)
   PCOPY <DRIVESPEC> <DRIVESPEC>
    PDEL <DRIVESPEC>
```

```
 PROTECT <FILESPEC> [D][W][C][X]
    PUSH <FILESPEC>
    QUIT (terminates emulator session via hardwired key, saves vdisks)
  RENAME <FILESPEC> <FILESPEC>
 REORDER <DRIVESPEC>
     RUN <FILESPEC>
    SAVE <HHHH 0000-BFFF> <HHHH 0000-BFFF>
SAVE.LOW <HHHH C000-FFFF> <HHHH C000-FFFF>
   SPLIT <FILESPEC>
  TTYSET [BS=HH] [BE=HH] [DL=HH] [EL=HH] [DP=DD]
  TTYSET [WD=DD] [NL=DD] [TB=HH] [EJ=DD] [PS=Y|N]
  TTYSET [ES=HH]
  UPDATE <FILESPEC>
  VERIFY [ON] [OFF]
 WIDECAT <DRIVESPEC>
    XOUT <DRIVESPEC>
       Y <flex command> (answers all questions with Y)
    YEAR <YY>

 STYHLP1 (help file for STYLUS/EDIT)
 STYHLP2 (help file for STYLUS/EDIT)
 STYHLP3 (help file for STYLUS/EDIT)
 STYHLP4 (help file for STYLUS/EDIT)
 STYHLP5 (help file for STYLUS/EDIT)
 STYHLP6 (help file for STYLUS/EDIT)
 PRINT.SYS (for printer driver)
 HELPFILE.DIR (help for Flex commands, via HELP command)
 BASIC.DIR    (help for BASIC, via HELP command)
 EXAMPLE.DIR  (shows you how to write your own help databases)
```

Appendix C - Changes from Version 1.0

v1r33, May 31st, 1996 (Win95 version)

— Potential initialization bug found & fixed. Could have resulted in screen display beginning with unrefeshed desktop region(s). Would only have been a cosmetic problem, but ugly. Chalk it up to my being new to Win95 programming. :-(

— Month calculation in "About" dialog was wrong.

v1r32, May 27th, 1996 (Win95 version)

— Menu switch for auto-setting of flex date added (so you can set any date; otherwise, the system date will constantly override yours, and if you're trying to "fool" some software, you'll fail). When checked, flex date is set once a minute. When not checked, flex date is set on boot only.

— Formatted Microsoft Word 6 document now included in distribution.

— Changed names of DRIVE0 and DRIVE1 to DISK0 and DISK1 for consistency

v1r31, May 21st, 1996 (Win95 version)

— Menu switch for capital letter inversion feature added

v1r30, May 21st, 1996 (Win95 version)

— Rebuilt emulator as Win32 application, instead of console. Subsequent changes:

- Windows Menu added:
  - Exit Emulator (exit to Win95, saves VDISK changes)
  - Abort Emulator (exit to Win95, does NOT save VDISK changes!)
  - RESET 6809 (enters PSYMON monitor)
  - Flush virtual disks
  - Switchable thread release on check ACIA for input character
- Display is more consistent now, as I wrote new terminal emulator
- ASCII code 7 (BELL) in emulator now generates Win95 "beep".
- More friendly for multitasking, shares as per Win95 guidelines
- ANSI color control is no longer available. Only SOROC intensity supported.
- COLORS.BAS changed to show switch of intensity only
- ANSI.SYS no longer required in CONFIG.SYS file
- Updated README.TXT documentation considerably

v1r23, May 18th, 1996 (All versions)

— Added "FILETEST.CMD" to drive0 file; from original FLEX disk on 6809 system

v1r22, May 16th, 1996 (Win95 version)

— Enhanced SOROC emulator to accept IBM ANSI color control strings

— Added "COLORS.BAS" to DRIVE1 to demonstrate use of ANSI COLORS

— Defaulted Flex to 0 NULLS, which fixes cosmetic issue on startup

— Added version identification to the software

v1r21, May 14th, 1996 (Win95 version)

— Enhanced INJECT to strip line feeds

— Enhanced DISTILL to add line feeds on output

— Fixed problem with padding end of file to zeros in INJECT

v1r20, May 12th, 1996 (Win95 version)

— Ported Flex Emulator code Win95 using Visual C++ version 4.0
— Ported FDIR to win95
— Ported INJECT to win95
— Ported DISTILL to win95
— Ported MAD to win95
— "Flipped" upper & lower case, as Flex is mainly an upper case OS
— Added command to re-flip the case defaults: FCASE.CMD
— No menu controls available in Microsoft's "QuickWin" 4 Environment
— FCASE.CMD added to flip state of upper case inversion
— FLUSH.CMD added to rewrite vdisks immediately without exit
— QUIT.CMD added for controlled exit from Flex emulation (replaces menu)
— ABORT.CMD added for exit without rewrite of vdisks (replaces menu)
— Soroc IQ-140 emulation rebuilt to compensate for weak ANSI.SYS on PC

v1r20, April 28th, 1996 (Amiga version)

— RESET command available from Amiga Menu
— All illegal opcodes now translate to $3F (SWI)

v1r10, March 26th, 1994 (Amiga version)

— Emulation runs about 2x as fast as v1r0
— Disks are now kept in memory, rewritten ONLY if you change them
— Critical (for Stylus) Soroc IQ140 terminal emulation operating
— Distribution includes Stylus ("EDIT") and help files
— Distribution includes RENAME command (left out by mistake in 1.0)
— Distribution includes Excellent "SLEUTH" dissassembler
— Distribution includes HELP.CMD and helpfiles (.DIR files)
— Distribution includes FILESORT command by Brian Bailey
— Distribution includes PUSH command (left out by mistake in 1.0)
— Distribution includes UPDATE command
— Distribution includes BASIC command (helpfile, too!)
— Distribution includes FREEMAP command
— EDIT command (line editor) renamed to LEDIT

v1r00, March 25th, 1994 (Amiga version)

— Original Release