UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR. 403/8

FIFTH COLLEGE ON MICROPROCESSORS: TECHNOLOGY AND APPLICATIONS IN PHYSICS

2- 27 October 1989

Introduction to FLEX

C. VERKERK
CERN, DD Division, Geneva, Switzerland

<u>These notes are intended for internal distribution only.</u>

INTRODUCTION TO FLEX

1. INTRODUCTION TO ROSY

We have seen in the introductory lectures that a computer needs a program to do some useful work. In a turnkey system, this program will start up automatically when power is switched on and will expect no or little operator intervention. This is for instance the case for a numerically controlled machine tool: the operator intervention is limited to pushing a few buttons with well-defined functions.

For a general purpose computer, for which we want to develop programs ourselves, or on which we want to run different programs of our choice, we need a means of communicating with the machine. We want to be able to type commands on our terminal, receive output on the terminal or on a printer, edit a text, translate a program etc. One of the tasks of an Operating System is to provide this communication.

An operating system is in general a large set of programs, normally stored on a disk or another permanent storage medium. It often relies for the simple functions such as input/output from/to a terminal on another program, which resides permanently in the computer, in Read-Only Memory (ROM). This program is usually called the Monitor Program, or simply Monitor. When power is switched on, or the Reset button pushed, this Monitor program will automatically start to execute. It will display a 'prompt' on the screen to indicate that it has come to life. It will then wait for input from the keyboard.

A Monitor can be very simple and only be able to accept and execute primitive commands. A command to inspect the contents of memory locations and to modify those contents, together with a command to start execution of a user program at a given address, is the strict minimum a monitor should have. As ROMs are cheap and compact, the tendency is to make monitors more complex and thus more useful and also more user-friendly.

The ROSY Monitor, installed on our development stations is an example of a more elaborate monitor. It was designed and written by Allessandro Marchioro and Wolfgang von Rüden of CERN. It has a large number of useful commands, and it is very user-friendly. For instance command names may be shortened, as long as they remain unambiguous. So, instead of typing: MEMORY-MODIFY, it is sufficient to type M-M. M-L is not a valid command, as it could mean MEMORY-LIST as well as MEMORY-LOAD. We must type M-LI or M-LO to distinguish between the two.

Some of the ROSY commands need (numerical) parameters. When the user forgets to specify the parameter(s), the monitor will ask for them. Some parameters are compulsory, others just specify options and may be left out. The monitor will in that case substitute default values. A HELP command is available to show the user the commands available and the parameters that are required.

In addition to the commands, the ROSY Monitor contains a number of routines which may be called from a user program. These routines perform input/output functions and number conversion functions mainly. They are of course extremely useful, for they take the burden of programming the input/output away from the user, who can concentrate his efforts on translating his algorithms into the programming language. When he wants to print a result, he can do this simply by making a so-called Monitor Call, after he has set up the necessary parameters. All Monitor Calls available are described in the FLOXY Manual, of which you have a copy. For instance, when I want to display on my terminal a 16-bit number, residing in memory (the result of a calculation, say), I load the address of the memory location in the X register, I set up the B register to indicate how I want the number displayed (in decimal or in hexadecimal) and I perform the Monitor Call number 35. The Monitor call is nothing else than a SWI instruction, followed by a single byte number, which indicates which call I want to perform (see the FLOXY manual). So in my program I would write:

```
          SWI          this is the Monitor Call
          FCB 35       this indicates which one I want (the general print routine
                       in this particular case).
```

When you become a bit more advanced, you will see that facilities exist to write the monitor calls in a more digestible way, using mnemonics instead of the sequence of an instruction and an assembler directive. To start the first exercises on ROSY, it is sufficient to remember the very important monitor call that returns to the Monitor program from the user program. When your program has the following sequence as the last thing it executes, it will return properly to ROSY, the prompt ROSY > will be displayed and you may type in another command. If you don't finish the execution of your program with

```
          SWI          monitor call to return to ROSY
          FCB 0
```

the strangest things may happen!

2. ROSY COMMANDS

The ROSY commands can be classified in the following categories:

- **Inspect/modify memory locations:**
  They are MEMORY-COMPARE, MEMORY-DUMP, MEMORY-FILL, MEMORY-LIST, MEMORY-LOAD, MEMORY-MODIFY, MEMORY-VERIFY, COPY, SEARCH.
  There most simple expressions are M-C, M-D, M-F, M-LI, M-LO, M-M, M-V, C and SEA.
- **Inspect/modify register contents:**
  They are REGISTER-DISPLAY and REGISTER-MODIFY. The stripped down versions are R-D and R-M.
- **Debugging aids:**
  These comprise the breakpoint commands BREAKPOINT-CLEAR and BREAKPOINT-SET and TRACE. SYMBOLS and DECODE are other debugging aids. Shorthand notation is B-C, B-S, TR, SY and DEC. Their use has been explained in the lectures. Briefly: Execution of a program stops when a breakpoint is reached; the contents of all registers is then displayed. The two breakpoint commands allow you to place and to remove breakpoints. TRACE will execute a specified number of instructions, starting at the current position of the program counter, one by one and display after each instruction the contents of all registers. DECODE disassembles code stored in memory and displays a listing in assembly language.
- **Execution and Input/output commands:**
  These are: GO, RESTART, MONITOR-EXECUTE, SLAVE, TMODE and FLEX. Stripped down: G, RES, M-E, SL, TM and F. GO, followed by an address will launch execution of the program at that address, RESTART allows you to start again, MON-EXEC makes it possible to make monitor calls directly from the terminal instead of a program. SLAVE and TMODE declare devices other than the terminal to be used for input/output. (see the FLOXY manual). FLEX is the command which will bring in the FLEX operating system, on which more below.
- **Miscellaneous:**
  Here we have HELP, DATA-CONVERT and SET-NUMBER-BASE. Shorthand is H, D-C, and S-N-B. DATA-CONVERT allows you to convert numbers from one base to another,

S-N-B tells ROSY the number base in which input/output has to take place (on default this is hexadecimal).

The Monitor Calls can also be classified according to their functions. The complete list with the detailed description of each call is given in the manual. It should be noted that these calls are of very general use: a user program may have recourse to them, but also the FLEX operating system and even the ROSY monitor itself make largely use of them. There are monitor calls for doing the following things:

- **Character or text input/output.**
- **Number input/output.**
- **Output of error messages.**
- **Redirection of input and output to other devices.**
- **Setting of vectors and other system parameters.**
- **Setting-up and handling of input/output buffers.**
- **Some memory operations, such as filling and searching.**
- **Task activation and de-activation.**
- **Timing routines.**

3. THE FLEX OPERATING SYSTEM.

From the brief description above you see that the ROSY Monitor is very powerful and that it allows you to do many things. You have also got a feeling of its power during your last exercise. There are however a number of things the ROSY Monitor cannot do and for which an operating system is needed. We have chosen to run the FLEX Operating System on top of the ROSY Monitor (thus the name FLOXY). FLEX is an operating system designed by Technical Systems Consultants for the 6809 processor. It is not the most powerful operating system available for this processor, but it is simple and easy to learn. It has sufficient functionality to be useful for all practical purposes on single user systems.

FLEX provides the functions that ROSY does not have and without which it would be impossible to do any useful work. It gives us access to:

- **A File System.**
  Files are stored on floppy disk and can be accessed by name.
- **An Editor.**
  This allows you to type in your programs and to correct them.
- **An Assembler.**
  The Assembler translates programs written in Assembly Language into machine code.
- **Other high-level language processors.**
  There is a Pascal compiler and a Basic interpreter available.
- **Other Utility Packages.**
  A sophisticated debugger and a disassembler are examples.

FLEX makes use of ROSY routines to do the following:

- **Character input/output from/to the terminal,**
- **Reading from and writing to floppy disk,**
- **Task activation for "printer spooling".**
  This last term means that you can continue to use FLEX commands from your terminal, while the system is simultaneously printing a file on the printer. You therefore do not have to wait until a long listing has been entirely printed, but you may immediately go on doing other things, for instance assemble another program.

It is very easy to get the FLEX system going:

- **Switch on the station (or push reset)**
- **Wait for the prompt ROSY > to appear,**
- **Insert a system disk in drive 0,**
- **Then type on your keyboard: FLEX or simply: F,**
- **The system will start and ask for the date.**
- **Type in the date in the form: MM DD YY (e.g. 09 25 86),**
- **After a few seconds the prompt + + + will appear, showing that**
  showing that FLEX is now fully operational and waiting for your f first command.

We should make a little aside here on 'prompts'. A prompt always indicates that the program which has control of your machine is waiting for some input from you. Different programs use different prompts, so you will always be able to tell which program is running (in case you forgot...). As you will often be switching forward and backward between ROSY and FLEX in the coming weeks it is important to keep in mind that ROSY will not understand the meaning of FLEX commands and vice-versa. So, always watch the prompt:

- **If the monitor has control:** ROSY >
- **IF FLEX has control:** + + +
- **When you are editing:** #
- **When the Assembler is running:**

Figure 1 shows your screen after you have started FLEX and executed a command. Note the different prompts. Figure 2 shows a little program, first in the form it has been typed in, then after it has been assembled by the assembler. The output listing is nicely arranged in columns, whereas the input lines start either in column 1 or 2, depending on the presence of a label field or not.

4. FLEX COMMANDS

Note that on a 2-drive station, the FLEX system disk must be in drive 0. To inspect what is on this disk, there are three different commands available. In what follows, the FLEX prompt is also shown, followed immediately by what you have to type in.

+ + + FILES⇐ **(⇐ means the carriage return key on the keyboard)**
+ + + CAT⇐ **(you may also type FILES,0 or FILES 0)**
+ + + DIR⇐ **(typing CAT 1 or DIR 1 will show the contents of disk 1)**

Examples of the output produced by these commands are given in Figure 3. Note that the DIR command gives the most information about the contents of your disk. It not only gives the names of all the files, but also their size (in units of 1 sector, equivalent to 256 bytes), their creation date and their physical location on the disk. If you want to know if there is still space left on your disk for another file, use the following command:
+ + + FREE⇐ **(this will tell you how many sectors are still usable)**

4.1 File Specification

A complete file specification consists of three parts: the number of the drive where the disk containing the file is inserted, the filename itself and an extension. The filename must start with a letter, may not exceed 8 characters and may not contain characters other than the letters A-Z, the digits 0-9, the underscore _ and the minus sign -. Examples of valid file specifications are:
1.MYFILE.BIN
0.SNOOPY7.TXT etc.

In many cases one or more of the three fields may default to preset values:

- The drive number may often be left out. The system will find the file in most cases. When you have on both drives different files, but having the same name, you better specify the drive number!
- In some cases the name is not needed. This is for instance the case for the assembler, which will generate a binary file with the same name as the input file, but with another extension.
- The extension may often be left off. The editor for example expects a text file (extension TXT) as input, the loader a binary file (BIN).

At present the following extensions may be used, or are used by some specific programs: .SYS .CMD .TXT .BIN .OUT .SYM .HLP .BAK

Most FLEX commands reside on disk (exceptions are MON and GET) in the form of a file with the extension .CMD. The complete command line to list the file SNOOPY.OUT which resides on disk 1 would take the form:

```
     + + + 0.LIST.CMD,1.SNOOPY.OUT⇐
```

The first part tells FLEX which command to execute, the second indicates the file on which the operation should be performed. Fortunately, in the majority of cases this command and similar commands can be shortened:

```
     + + + LIST SNOOPY⇐
```

Note that the separation character is either 'a space' or 'a comma'.

4.2 Useful Commands.

The most used FLEX commands are listed below. For details of their function, see the FLEX Manual, which is attached to the work station. In what follows, parameters are indicated between < and >, when they are optional they are placed between [and]. A <file spec> is either a complete file specification or a reduced one.

```
     + + + FILES⇐
     + + + CAT⇐
     + + + DIR⇐
     + + + MON⇐                                  gets you back to ROSY
     + + + LIST,<file spec>[,<line range>][,N]⇐  lists N copies
                                                 of lines of a file.
     + + + PRINT,<file spec>⇐                    prints file, using spooling.
     + + + COPY,<file spec1>,<file spec2>⇐       copies file 1 to file 2
     + + + SAVE,<file spec>,<begin addr>,<end addr>[,<tr.addr>]⇐
                                                 this saves the memory locations from
                                                 'begin address' to 'end address' as a file.
     + + + DELETE,<file spec>⇐                   this deletes a file
     + + + RENAME,<file spec1>,<file spec2>⇐     gives a new name to file 1
     + + + P,<command>⇐                          <here command can be any
                                                 FLEX command which would produce
                                                 output to the screen. The output
                                                 now goes to the printer instead.
     + + + O,<file spec>,<command>⇐              the screen output normally produced
                                                 by <command> now goes to the file.
     + + + GET,<file spec>⇐                      loads the binary file into memory
     + + + LOAD,<file spec>⇐                     same as GET, but loads the symbol
                                                 table as well.
     + + + ASM,<file spec>[,various options]⇐   calls the assembler
                                                 to translate the program in the file.
     + + + EDIT,<file spec>[,various options]⇐  calls the editor, uses file.
```

Some examples of the use of these commands are:

```
  + + + P,LIST,MYFILE.TXT⇐             (this is in fact equivalent to
                                       PRINT,MYFILE⇐, but with
                                       a subtle difference.
  + + + O,CONTENTS,DIR,1⇐              this will save output of DIR in a
                                       file named CONTENTS.
  + + + LIST,CONTENTS⇐                 this will then list it.
```

The effect of the two last commands together is the same as the effect of DIR,1 alone. Again there is a small difference. Which?

5. THE EDITOR.

The editor will allow you to type in text, save it on disk and, if necessary, modify it. The text is usually a program, but it does not need to be. It can be a letter, a poem or a list of telephone numbers.

The editor (which is itself a program, of course), is invoked with the command:

```
  + + + EDIT,<file spec>.
```

For example: + + + EDIT,TEST1.
Two cases can present themselves:
1) The file specified does already exist. The Editor will then read it into its buffer space and then present its prompt: #
2) The file specified does not yet exist. In that case the Editor will create a new file with this name and respond with:

```
  New File
  1.00 =                               now you can start typing!
```

After you have typed a first line, and ended it with a carriage return, the Editor will ask for the next line with:

```
  2.00 =                               etc.
```

When you have finished typing new lines, you must type a # in reply to the line number prompt:

```
  389.00 = #                           the # sign is what you typed.
  #                                    this was typed by the editor!
                                       it now waits for a command!
```

The FLEX Editor provided on your system disk is line oriented. It works with the concept of the 'current line'. So if you type the letter 'P' in reply to the prompt, the current line will be displayed on the screen. You can make any line you want the current line by preceding the editor command with the line number:

```
  # 14P                                will display line 14.
  # 14P5                               will display 5 lines from 14
                                       onward: lines 14, 15, 16, 17 and 18
  # ∧P!                                will list all lines.
```

In this last command, the ∧ sign will make the first line the current line. The ! mark tells the editor to continue till the last line in the buffer.
Other commands which work on a single line (the current line in fact) can be modified in the same way:

```
  # 14D                                will delete line 14
  # 14D5                               will delete lines 14 to 18.
```

I would not recommend to try the command ∧D!, unless you are a good typist...

The editor, just as ROSY, accepts shorthand notation for its commands, as long as the meaning is unambigious. Other important editor commands with their shorthand notation are:

# RENUMBER	or REN	renumbers the lines, using integers only.
# BOTTOM	or B	moves the current line to the end of the file.
# TOP	or T	moves it to the first line.
# LOG	or L	logs out from editor and returns to FLEX.
# STOP	or S	does the same as LOG
# FIND	or F	finds a string of characters.

This is a very useful command, which allows you to locate a string of characters in your text.

# F/my nice string/	will find this character string in the text (if it exists) and list the line where it occurred.

There are many variations of the FIND command possible, see the Editor Manual for details.

The commands above worked on the text file as a whole. Another series of commands allow you to edit the text, e.g. modify it, delete parts and add others.

# APPEND/string/	will add something to the end of the file.
# CHANGE/string1/string2/	changes string1 into string2
# DELETE	this is the D command already seen.
# INSERT	inserts new lines below the current line.
# OVERLAY	does makes it possible to change individual characters in a line. See manual.
# PRINT	this is the P command already mentioned
# REPLACE	this is equivalent to D followed by I

For options, refinements and variants of these commands, see the manual! Also study the example session in Figures 6 and 7.

Supposing that line 14 contained:
HERE ADDA NUMBER some comment,
then the command: 14C/HERE/THERE/ would change this into
THERE ADDA NUMBER some comment.
Executing the INSERT command, the following would happen:

```
# 14I
14.10 = *This is an extra comment          the *This... was typed by us!
14.20 = *and another comment
14.30 = *
14.40 = #                                  we typed #
#                                          the editor now answers with #.
```

When you have finished working with the Editor, you get out of it by typing S as reply to the # prompt. The newly created or edited file will be saved on disk under the name you gave originally when you invoked the editor. If this file existed already before, a back-up copy of the original, unmodified file is kept. The back-up copy will have the same name, but the extension is .BAK.

6. THE ASSEMBLER

The Assembler, which will translate programs written in Assembly Language into binary Machine Code, is invoked with the following FLEX command:

+ + + **ASM, < file spec > [,various options]** **for options, see the manual!**

The Assembler will be loaded and start running. It may ask you a few questions, and then the disk will start spinning and your source program will be read. The questions it may ask you are to obtain your permission to delete old files, if they exist (e g. if they have been generated during a previous run of the Assembler). So, for example, if you type.

+ + + **ASM,MYPROG** **this will use MYPROG.TXT as source**

then the Assembler will produce a number of things:

- **You will get an output listing on your screen**
- **the Assembler will produce a binary file on disk: MYPROG.BIN**
- **also the symbol table will be written on disk: MYPROG.SYM**

In case you want the listing of your program printed on paper, then there are two possible ways of doing this:

1) + + + **P,ASM,MYPROG** **this redirects output to the printer**

2) + + + **O,MYPROG.OUT,ASM,MYPROG** **output listing goes to file MYPROG.OUT**
+ + + **PRINT,MYPROG.OUT** **and this will then print it.**

When you have errors in your program, the Assembler will tell you so. The error messages are interspersed with the output listing. They may seem rather cryptic, especially for a beginner. The manual, or an instructor will help you in case of trouble. The Assembler puts another message at the end of the output listing

47 ERROR(S) DETECTED **this number is generally exaggerated**
a single mistake may generate several errors.

The Assembler then lists the Symbol Table, which is also written to disk, for later use by the ROSY Monitor.

Figure 8 shows an example of a run with the assembler.

7. RUNNING YOUR PROGRAM

Once you have assembled your program and you got the message that no errors were detected, you can try to run it. The fact that the Assembler did not find any error does not mean that your program will run without problems. The Assembler cannot detect logical errors or guess what you wanted to obtain as a result and correct the mistakes you made when you translated your ideas into a program! These errors you will find when you run the program.

To run your program, the first thing to do is to load it into memory:

+ + + **LOAD.MYPROG** **this will load MYPROG.BIN from disk**

The symbol table will be read also and stored in memory. This allows ROSY to know the names of your variables and labels and to know the addresses that are assigned to them. You may now use symbols instead of absolute addresses in hexadecimal when you debug your program with the help.of the ROSY debugging commands. You noticed that debugging is done with the help of the ROSY Monitor, so after your program has been loaded into memory, you should call the monitor:

+ + + **MON** **this will give ROSY > on the screen**

Now ROSY has control and you may do whatever is needed for debugging:

- **Set breakpoints**
- **inspect memory**
- **modify memory**
- **set registers**
- **trace**
- **etc. etc.**

After you have set your breakpoints, or made patches to your program, you can start executing it by typing:

```
    ROSY> GO START                    (if your entry point is called START)
```

or

```
    ROSY> GO 2100                     (if the program entry is at hex address 2100)
```

or

```
    ROSY> GO BEGIN
```

or something of that kind.

Now only one thing remains to be done:

KEEP YOUR

FINGERS

CROSSED !

```
ROSY >F
6809 FLEX V3.01

DATE (MM,DD,YY)? 10 10 85

**********************************************************************
*                                                                    *
*           INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS             *
*                                                                    *
*                    Microprocessors Laboratory                      *
*                                                                    *
*                                                                    *
*                              WELCOME                               *
*                                                                    *
*                              to the                                *
*                                                                    *
*                 THIRD COLLEGE ON MICROPROCESSORS                   *
*                                                                    *
*                                                                    *
*                              TRIESTE                               *
*                                                                    *
*                  7 OCTOBER - 1 NOVEMBER 1985                       *
*                                                                    *
*                                                                    *
**********************************************************************

+++date
OCTOBER 10, 1985
+++list,pim
NUMBER EQU $10
POPO EQU $20
 ORG $100
START NOP
 LDA #$40
 SUBA #NUMBER
 STA DIFF
 ADDA #POPO
 STA DIFF+1
 SWI
 FCB 0
DIFF RMB 2
 END
```

START UP OF FLEX

EXAMPLES OF FLEX COMMANDS

FIG. 1.

A SMALL EXAMPLE PROGRAM

BEFORE ASSEMBLY

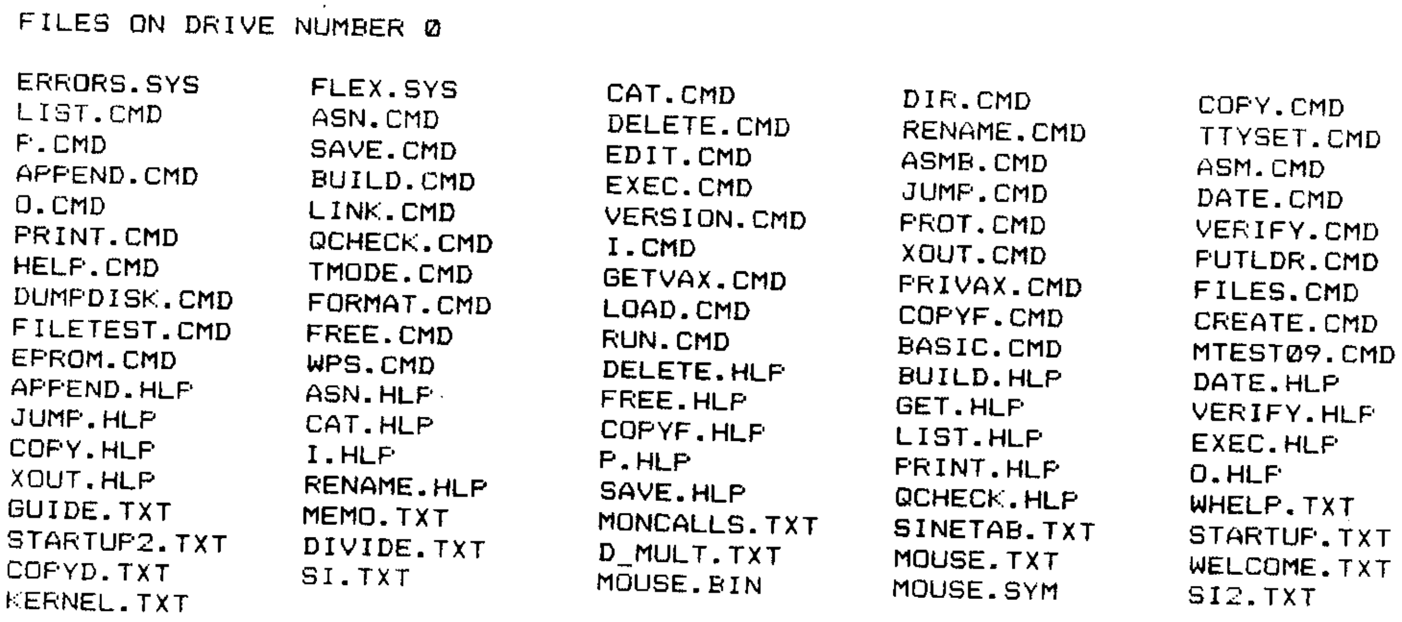
```
NUMBER EQU $1Ø
FOPO EQU $2Ø
 ORG $1ØØ
START NOP
 LDA #$4Ø
 SUBA #NUMBER
 STA DIFF
 ADDA #FOPO
 STA DIFF+1
 SWI
 FCB Ø
DIFF RMB 2
 END
```

AFTER ASSEMBLY

```
                 ØØ1Ø  NUMBER  EQU   $1Ø
                 ØØ2Ø  FOPO    EQU   $2Ø
 Ø1ØØ                          ORG   $1ØØ
 Ø1ØØ 12               START   NOP
 Ø1Ø1 86   4Ø                  LDA   #$4Ø
 Ø1Ø3 8Ø   1Ø                  SUBA  #NUMBER
 Ø1Ø5 B7   Ø1ØF                STA   DIFF
 Ø1Ø8 8B   2Ø                  ADDA  #FOPO
 Ø1ØA B7   Ø11Ø                STA   DIFF+1
 Ø1ØD 3F                       SWI
 Ø1ØE ØØ                       FCB   Ø
 Ø1ØF                  DIFF    RMB   2
                               END

Ø ERROR(S) DETECTED

SYMBOL TABLE:

DIFF   Ø1ØF   NUMBER ØØ1Ø   FOPO   ØØ2Ø   START  Ø1ØØ
```

FIG. 2

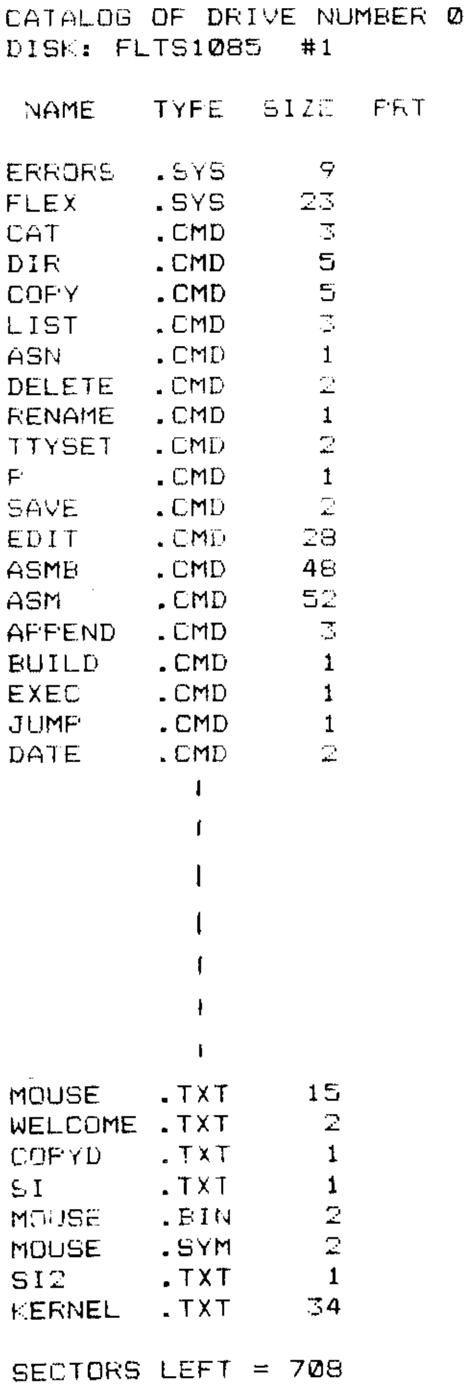
RESULT OF "FILES"



```
FILES ON DRIVE NUMBER Ø

ERRORS.SYS      FLEX.SYS        CAT.CMD         DIR.CMD         COPY.CMD
LIST.CMD        ASN.CMD         DELETE.CMD      RENAME.CMD      TTYSET.CMD
P.CMD           SAVE.CMD        EDIT.CMD        ASMB.CMD        ASM.CMD
APPEND.CMD      BUILD.CMD       EXEC.CMD        JUMP.CMD        DATE.CMD
O.CMD           LINK.CMD        VERSION.CMD     PROT.CMD        VERIFY.CMD
PRINT.CMD       QCHECK.CMD      I.CMD           XOUT.CMD        PUTLDR.CMD
HELP.CMD        TMODE.CMD       GETVAX.CMD      PRIVAX.CMD      FILES.CMD
DUMPDISK.CMD    FORMAT.CMD      LOAD.CMD        COPYF.CMD       CREATE.CMD
FILETEST.CMD    FREE.CMD        RUN.CMD         BASIC.CMD       MTESTØ9.CMD
EPROM.CMD       WPS.CMD         DELETE.HLP      BUILD.HLP       DATE.HLP
APPEND.HLP      ASN.HLP         FREE.HLP        GET.HLP         VERIFY.HLP
JUMP.HLP        CAT.HLP         COPYF.HLP       LIST.HLP        EXEC.HLP
COPY.HLP        I.HLP           P.HLP           PRINT.HLP       O.HLP
XOUT.HLP        RENAME.HLP      SAVE.HLP        QCHECK.HLP      WHELP.TXT
GUIDE.TXT       MEMO.TXT        MONCALLS.TXT    SINETAB.TXT     STARTUP.TXT
STARTUP2.TXT    DIVIDE.TXT      D_MULT.TXT      MOUSE.TXT       WELCOME.TXT
COPYD.TXT       SI.TXT          MOUSE.BIN       MOUSE.SYM       SI2.TXT
KERNEL.TXT
```

FIG. 3.

RESULT OF "CAT"



```
CATALOG OF DRIVE NUMBER Ø
DISK: FLTS1085   #1

 NAME    TYPE  SIZE  PRT

ERRORS  .SYS     9
FLEX    .SYS    23
CAT     .CMD     3
DIR     .CMD     5
COPY    .CMD     5
LIST    .CMD     3
ASN     .CMD     1
DELETE  .CMD     2
RENAME  .CMD     1
TTYSET  .CMD     2
P       .CMD     1
SAVE    .CMD     2
EDIT    .CMD    28
ASMB    .CMD    48
ASM     .CMD    52
APPEND  .CMD     3
BUILD   .CMD     1
EXEC    .CMD     1
JUMP    .CMD     1
DATE    .CMD     2
          |
          |
          |
          |
          |
          |
          |
MOUSE   .TXT    15
WELCOME .TXT     2
COPYD   .TXT     1
SI      .TXT     1
MOUSE   .BIN     2
MOUSE   .SYM     2
SI2     .TXT     1
KERNEL  .TXT    34

SECTORS LEFT = 7Ø8
```

FIG. 4.

RESULT OF "DIR"

```
DIRECTORY OF DRIVE NUMBER Ø
DISK: FLTS1Ø85  #1    CREATED: 1Ø-SEP-85

FILE#   NAME    TYPE  R  BEGIN   END   SIZE    DATE       PRT


    1  ERRORS  .SYS  R  Ø1-Ø1  Ø1-Ø9     9  1Ø-SEP-85
    2  FLEX    .SYS     Ø1-ØA  Ø1-2Ø    23  1Ø-SEP-85
    3  CAT     .CMD     Ø2-Ø1  Ø2-Ø3     3  1Ø-SEP-85
    4  DIR     .CMD     Ø2-Ø4  Ø2-Ø8     5  1Ø-SEP-85
    5  COPY    .CMD     Ø2-Ø9  Ø2-ØD     5  1Ø-SEP-85
    6  LIST    .CMD     Ø2-ØE  Ø2-1Ø     3  1Ø-SEP-85
    7  ASN     .CMD     Ø2-11  Ø2-11     1  1Ø-SEP-85
    8  DELETE  .CMD     Ø2-12  Ø2-13     2  1Ø-SEP-85
    9  RENAME  .CMD     Ø2-14  Ø2-14     1  1Ø-SEP-85
   1Ø  TTYSET  .CMD     Ø2-15  Ø2-16     2  1Ø-SEP-85
   11  P       .CMD     Ø2-17  Ø2-17     1  1Ø-SEP-85
   12  SAVE    .CMD     Ø2-18  Ø2-19     2  1Ø-SEP-85
   13  EDIT    .CMD     Ø2-1A  Ø3-15    28  1Ø-SEP-85
   14  ASMB    .CMD     Ø3-16  Ø5-Ø5    48  1Ø-SEP-85
   15  ASM     .CMD     Ø5-Ø6  Ø6-19    52  1Ø-SEP-85
   16  APPEND  .CMD     Ø6-1A  Ø6-1C     3  1Ø-SEP-85
   17  BUILD   .CMD     Ø6-1D  Ø6-1D     1  1Ø-SEP-85
   18  EXEC    .CMD     Ø6-1E  Ø6-1E     1  1Ø-SEP-85
   19  JUMP    .CMD     Ø6-1F  Ø6-1F     1  1Ø-SEP-85
   2Ø  DATE    .CMD     Ø6-2Ø  Ø7-Ø1     2  1Ø-SEP-85
   21  O       .CMD     Ø7-Ø2  Ø7-Ø3     2  1Ø-SEP-85
   22  LINK    .CMD     Ø7-Ø4  Ø7-Ø4     1  1Ø-SEP-85
                              |
                              |
                              |
                              |
                              |
   72  MEMO    .TXT     ØD-18  ØF-1Ø    57  1Ø-SEP-85
   73  MONCALLS.TXT     ØF-11  ØF-17     7  1Ø-SEP-85
   74  SINETAB .TXT     ØF-18  ØF-1B     4  1Ø-SEP-85
   75  STARTUP .TXT     ØF-1C  ØF-1C     1  1Ø-SEP-85
   76  STARTUP2.TXT     ØF-1D  ØF-1D     1  1Ø-SEP-85
   77  DIVIDE  .TXT     ØF-1E  ØF-2Ø     3  1Ø-SEP-85
   78  D_MULT  .TXT     1Ø-Ø1  1Ø-Ø2     2  1Ø-SEP-85
   79  MOUSE   .TXT     1Ø-Ø3  1Ø-11    15  1Ø-SEP-85
   8Ø  WELCOME .TXT     1Ø-12  1Ø-13     2  1Ø-SEP-85
   81  COPYD   .TXT     1Ø-1A  1Ø-1A     1  17-SEP-85
   82  SI      .TXT     1Ø-1B  1Ø-1B     1  17-SEP-85
   83  MOUSE   .BIN     1Ø-16  1Ø-17     2  1Ø-SEP-85
   84  MOUSE   .SYM     1Ø-18  1Ø-19     2  1Ø-SEP-85
```

FIG. 5.

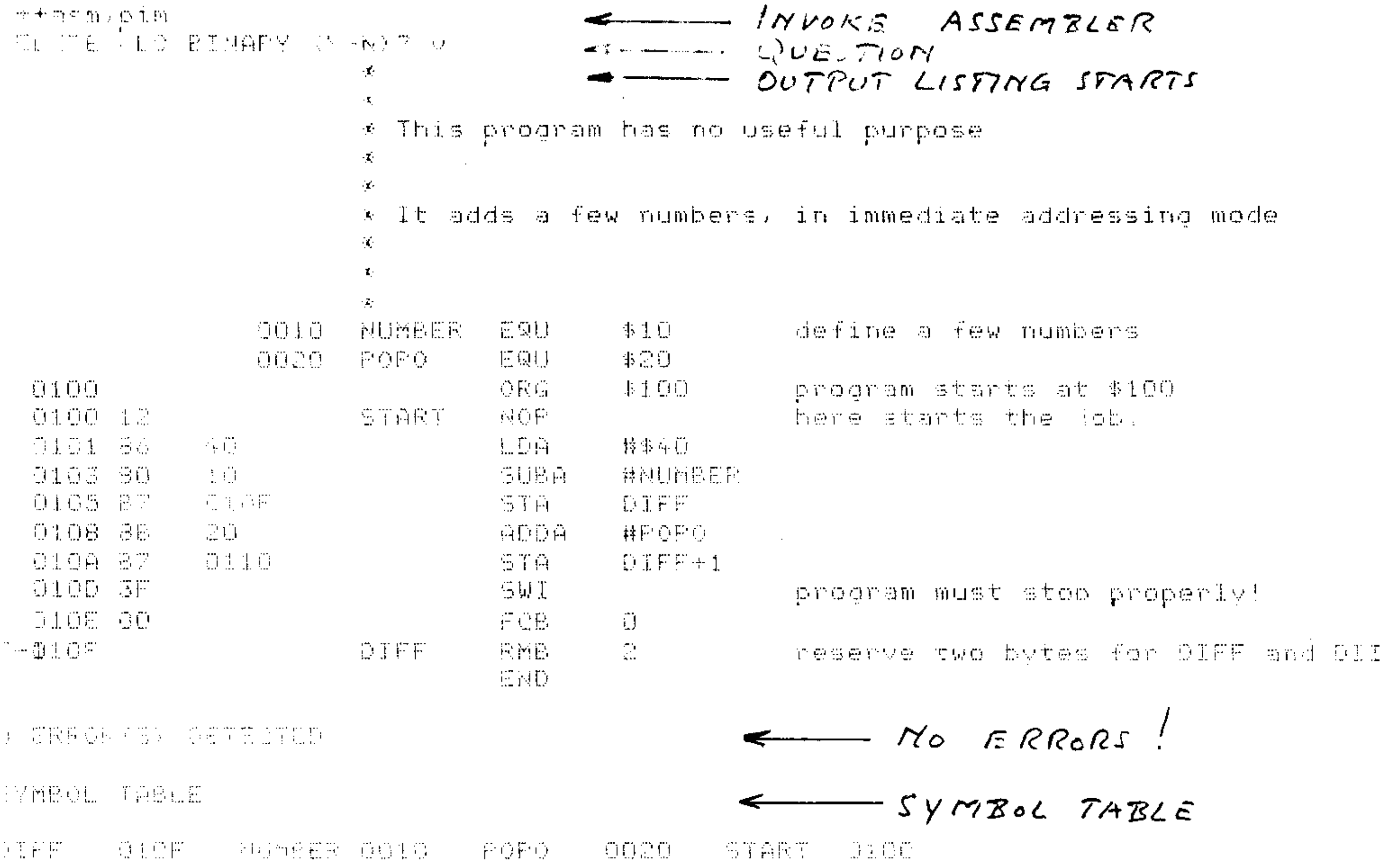
EXAMPLE OF AN EDIT SESSION.

```
++edit,pim                                  <------ EDIT COMMAND
DELETE BACKUP FILE (Y-N)? y                 <------ EDITOR ASKS QUESTION
#^p!                                        <------ LIST WHOLE FILE
    1.00=NUMBER EQU $10
    2.00=POPO EQU $20
    3.00= ORG $100
    4.00=START NOP
    5.00= LDA #$40
    6.00= SUBA #NUMBER
    7.00= STA DIFF
    8.00= ADDA #POPO
    9.00= STA DIFF+1
   10.00= SWI
   11.00= FCB 0
   12.00=DIFF RMB 2
   13.00= END
#0i                                         <------ INSERT BEFORE LINE 1
      .10= TTL FASULO                       <------ EDITOR TYPES .10=
      .20=*                                         USER TYPES THE LINE.
      .30=*
      .40=* This program has no useful purpose
      .50=*
      .60=*
      .70=* It adds a few numbers, in immediate addressing mde
      .80=* *
      .90=*
    1.00=*
    1.10=#                                  <------ TYPING # STOPS THE EDITOR
SOME LINES RENUMBERED                               FROM ASKING MORE LINES.
    1.00=*
#renumber                                   <------ ANOTHER EDITOR COMMAND
#^p!                                        <------ LIST ENTIRE FILE AGAIN
    1.00= TTL FASULO
    2.00=*
    3.00=*
    4.00=* This program has no useful purpose        COMMENTS ADDED
    5.00=*
    6.00=*
    7.00=* It adds a few numbers, in immediate addressing mde
    8.00=*
    9.00=*
   10.00=*
   11.00=NUMBER EQU $10
   12.00=POPO EQU $20
   13.00= ORG $100
   14.00=START NOP
   15.00= LDA #$40
   16.00= SUBA #NUMBER
   17.00= STA DIFF
   18.00= ADDA #POPO
   19.00= STA DIFF+1
   20.00= SWI
   21.00= FCB 0
   22.00=DIFF RMB 2
   23.00= END
#7c/mde/mode/                               <------ MAKE CORRECTIONS
    7.00=* It adds a few numbers, in immediate addressing mode
#11/1       ' define a few numbers
?                                           <------ EDITOR DOES NOT UNDERSTAND
#11/        ' define a few number           <------ AGAIN
?
#11 $10 $10 define a few numbers/           <------ NOW IT UNDERSTOOD
   11.00=NUMBER EQU $10 define a few numbers
```

FIG. 6.

```
#13o                                                     <-- MODIFY LINE 13 WITH "OVERLAY"
   13.00= ORG $100                                       <-- ORIGINAL
 OVERLAY              program starts at $100             <-- OVERLAY
   13.00= ORG $100    program starts at $100             <-- RESULT
#14c/$40/#40 here we are         at our iob/
"                                                        <-- EDITOR DOES NOT
                                                             UNDERSTAND.
#14p
   14.00=START NOP                          <-- PROVE (STRING $40 DOES NOT EXIST IN LINE 14)
#14c/    NOP/NOP here starts the iob?
   14.00=START NOP here starts the iob?          <-- NOW IT IS OK
#14o
   14.00=START NOP here starts the iob?          <-- GET RID OF ?
 OVERLAY                               .
   14.00=START NOP here starts the iob.
#p!                                              <-- LIST FROM CURRENT LINE
   14.00=START NOP here starts the iob.                               TO END
   15.00= LDA #$40
   16.00= SUBA #NUMBER
   17.00= STA DIFF
   18.00= ADDA #POPO
   19.00= STA DIFF+1
   20.00= SWI
   21.00= FCB 0
   22.00=DIFF RMB 2
   23.00= END
#20c/SWI/SWI program must stop properly!/ <-- ADD MORE COMMENTS
   20.00= SWI program must stop properly!     USING C AND O
#22o                                      <--
   22.00=DIFF RMB 2
 OVERLAY                reserve two bytes f  or . DIFF and DIIF+1
   22.00=DIFF RMB 2  reserve two bytes for DIFF and DIIF+1
#^p!                                                 FINAL LISTING
    1.00= TTL FASULO
    2.00=*
    3.00=*
    4.00=* This program has no useful purpose
    5.00=*
    6.00=*
    7.00=* It adds a few numbers, in immediate addressing mode
    8.00=*
    9.00=*
   10.00=*
   11.00=NUMBER EQU $10 define a few numbers
   12.00=POPO EQU $20
   13.00= ORG $100  program starts at $100
   14.00=START NOP here starts the iob.
   15.00= LDA #$40
   16.00= SUBA #NUMBER
   17.00= STA DIFF
   18.00= ADDA #POPO
   19.00= STA DIFF+1
   20.00= SWI program must stop properly!
   21.00= END
   22.00=DIFF RMB 2  reserve two bytes for DIFF and DIIF+1

#S                                                   <-- QUIT THE EDITOR
```

FIG. 7.



```
*+asm,pim
WRITE FILE TO BINARY (Y-N)? N
                         *
                         *
                         * This program has no useful purpose
                         *
                         *
                         * It adds a few numbers, in immediate addressing mode
                         *
                         *
                         *
                  0010  NUMBER   EQU    $10         define a few numbers
                  0020  POPO     EQU    $20
 0100                            ORG    $100        program starts at $100
 0100 12                START    NOP                here starts the job.
 0101 86    40                   LDA    #$40
 0103 80    10                   SUBA   #NUMBER
 0105 B7    010F                 STA    DIFF
 0108 8B    20                   ADDA   #POPO
 010A B7    0110                 STA    DIFF+1
 010D 3F                         SWI                program must stop properly!
 010E 00                         FCB    0
 010F                   DIFF     RMB    2           reserve two bytes for DIFF and DII
                                 END

0 ERROR(S) DETECTED

SYMBOL TABLE

DIFF    010F    NUMBER 0010    POPO    0020    START   0100
```

INVOKE ASSEMBLER

QUE.TION

OUTPUT LISTING STARTS

NO ERRORS!

SYMBOL TABLE

FIG. 8.

```
+++ LOAD, PIM.BIN
+++ MON
ROSY >
ROSY >decode 100 110
0100   12           NOP
0101   86    40     LDA    #$40
0103   80    10     SUBA   #$10
0105   B7    010F   STA    $010F
0108   8B    20     ADDA   #$20
010A   B7    0110   STA    $0110
010D   3F    00     MON    #$00
010F   30    50
ROSY >m-li 100 11F
```

FLEX COMMANDS "LOAD" AND "MON"

BACK IN ROSY

CHECK PROGRAM IS IN MEMORY

MEM-LIST

```
ADDR   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100  12 86 40 80 10 B7 01 0F 8B 20 B7 01 10 3F 00 30   ..@..........?.0
0110  50 C8 A5 EC 4A 44 56 44 56 44 56 17 1F F2 ED 4A   P...JDVDVDV....J
ROSY >m-mo 10F
010F 30 00
0110 50 00
0111 C8 .
ROSY >
ROSY >go      100

ROSY >m-li 100 11F

ADDR   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100  12 86 40 80 10 B7 01 0F 8B 20 B7 01 10 3F 00 30   ..@..........?.0
0110  50 C8 A5 EC 4A 44 56 44 56 44 56 17 1F F2 ED 4A   P...JDVDVDV....J
ROSY >+
```

SET TWO BYTES TO Ø.

RUN THE PROGRAM

RESULTS ARE HERE!

BACK TO FLEX

```
+++delete,pim.bin

DELETE "0.PIM.BIN" ? y
ARE YOU SURE? y
+++files,0
```

DELETE A FILE

IS THIS WHAT YOU WANTED?

CHECK THAT FILE HAS BEEN DELETED.

```
FILES ON DRIVE NUMBER 0

ERRORS.SYS      FLEX.SYS        CAT.CMD         DIR.CMD         COPY.CMD
LIST.CMD        ASN.CMD         DELETE.CMD      RENAME.CMD      TTYSET.CMD
P.CMD           SAVE.CMD        EDIT.CMD        ASMB.CMD        ASM.CMD
APPEND.CMD      BUILD.CMD       EXEC.CMD        JUMP.CMD        DATE.CMD
O.CMD           LINK.CMD        VERSION.CMD     PROT.CMD        VERIFY.CMD
PRINT.CMD       QCHECK.CMD      I.CMD           XOUT.CMD        PUTLDR.CMD
HELP.CMD        TMODE.CMD       GETVAX.CMD      PRIVAX.CMD      FILES.CMD
DUMPDISK.CMD    FORMAT.CMD      LOAD.CMD        COPYF.CMD       CREATE.CMD
FILETEST.CMD    FREE.CMD        RUN.CMD         BASIC.CMD       MTEST09.CMD
EPROM.CMD       WPS.CMD         DELETE.HLP      BUILD.HLP       DATE.HLP
APPEND.HLP      ASN.HLP         FREE.HLP        GET.HLP         VERIFY.HLP
JUMP.HLP        CAT.HLP         COPYF.HLP       LIST.HLP        EXEC.HLP
COPY.HLP        I.HLP           P.HLP           PRINT.HLP       O.HLP
XOUT.HLP        RENAME.HLP      SAVE.HLP        QCHECK.HLP      WHELP.TXT
GUIDE.TXT       MEMO.TXT        MONCALLS.TXT    SINETAB.TXT     STARTUP.TXT
STARTUP2.TXT    DIVIDE.TXT      D_MULT.TXT      MOUSE.TXT       WELCOME.TXT
COPYD.TXT       SI.TXT          MOUSE.BIN       MOUSE.SYM       SI2.TXT
[illegible].TXT PIM.TXT         PIM.BAK
```

PIM.BIN HAS GONE.

FIG. 9

START UP OF FLEX

```
ROSY >f
6809 FLEX V3.01

DATE (MM,DD,YY)? 10 10 85

******************************************************************
*                                                                *
*           INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS         *
*                                                                *
*                   Microorocessors Laboratorv                   *
*                                                                *
*                                                                *
*                            WELCOME                             *
*                                                                *
*                             to the                             *
*                                                                *
*                 THIRD COLLEGE ON MICROPROCESSORS               *
*                                                                *
*                                                                *
*                             TRIESTE                            *
*                                                                *
*                   7 OCTOBER - 1 NOVEMBER 1985                  *
*                                                                *
*                                                                *
******************************************************************

+++date
OCTOBER 10, 1985
+++list,pim
NUMBER EQU $10
POPO EQU $20
 ORG $100
START NOP
 LDA #$40
 SUBA #NUMBER
 STA DIFF
 ADDA #POPO
 STA DIFF+1
 SWI
 FCB 0
DIFF RMB 2
 END
```

EXAMPLES OF FLEX COMMANDS

FIG. 1.

A SMALL EXAMPLE PROGRAM

BEFORE ASSEMBLY

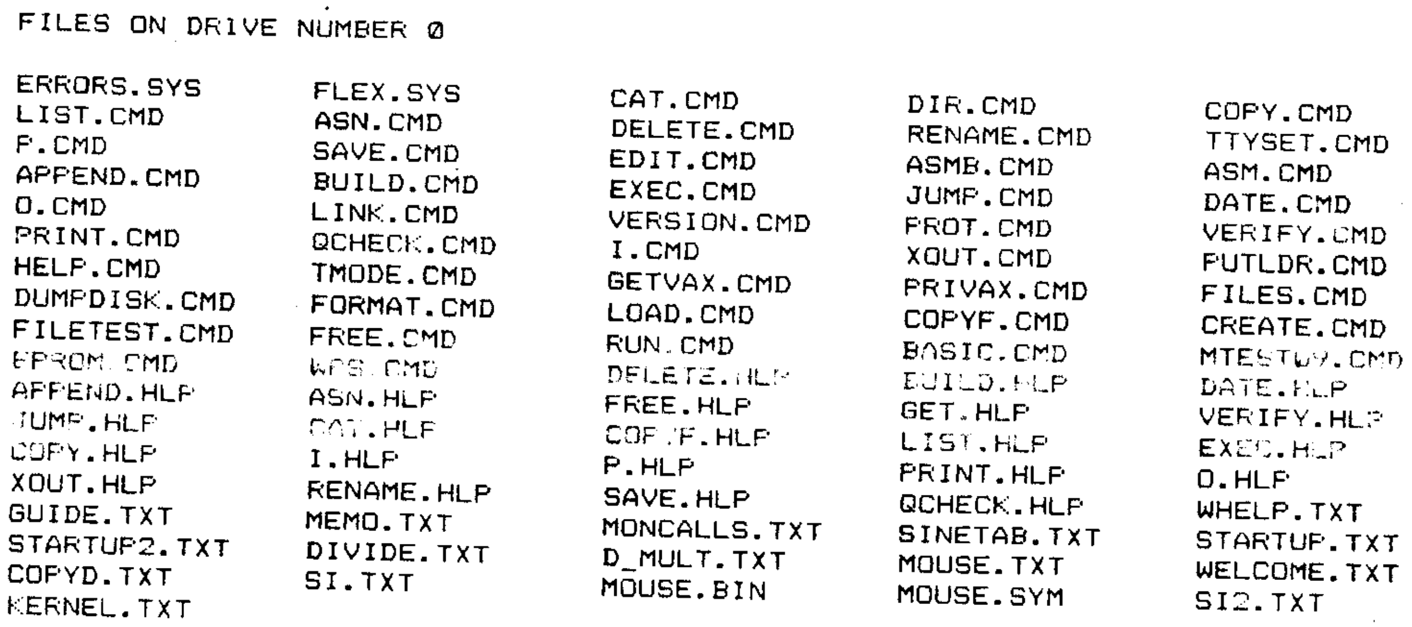
```
NUMBER EQU $10
POPO EQU $20
 ORG $100
START NOP
 LDA #$40
 SUBA #NUMBER
 STA DIFF
 ADDA #POPO
 STA DIFF+1
 SWI
 FCB 0
DIFF RMB 2
 END
```

AFTER ASSEMBLY

```
                0010  NUMBER  EQU    $10
                0020  POPO    EQU    $20
 0100                         ORG    $100
 0100 12              START   NOP
 0101 86   40                 LDA    #$40
 0103 80   10                 SUBA   #NUMBER
 0105 B7   010F               STA    DIFF
 0108 8B   20                 ADDA   #POPO
 010A B7   0110               STA    DIFF+1
 010D 3F                      SWI
 010E 00                      FCB    0
 010F                 DIFF    RMB    2
                              END

0 ERROR(S) DETECTED

SYMBOL TABLE:

DIFF   010F   NUMBER 0010   POPO   0020   START   0100
```

FIG. 2

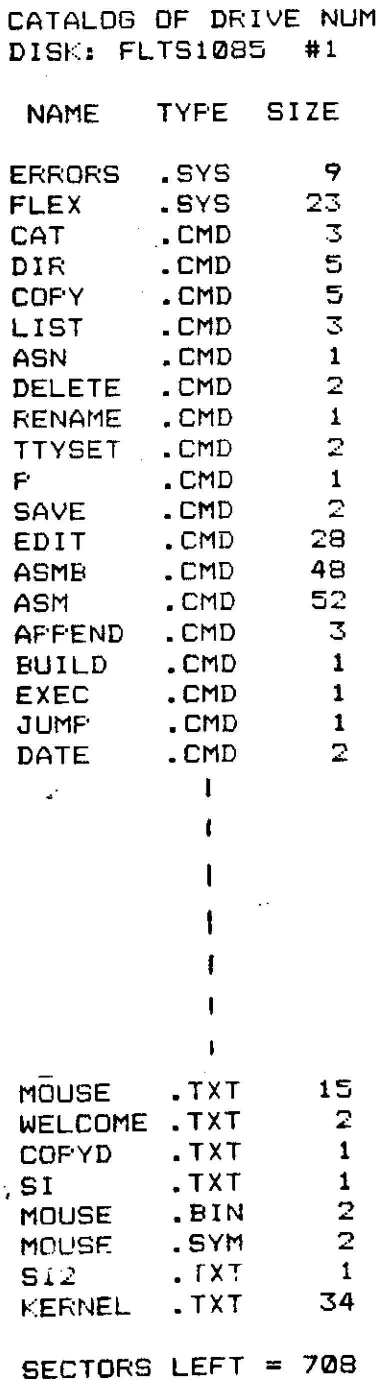
RESULT OF "FILES"



```
FILES ON DRIVE NUMBER 0

ERRORS.SYS     FLEX.SYS       CAT.CMD        DIR.CMD        COPY.CMD
LIST.CMD       ASN.CMD        DELETE.CMD     RENAME.CMD     TTYSET.CMD
P.CMD          SAVE.CMD       EDIT.CMD       ASMB.CMD       ASM.CMD
APPEND.CMD     BUILD.CMD      EXEC.CMD       JUMP.CMD       DATE.CMD
O.CMD          LINK.CMD       VERSION.CMD    PROT.CMD       VERIFY.CMD
PRINT.CMD      QCHECK.CMD     I.CMD          XOUT.CMD       PUTLDR.CMD
HELP.CMD       TMODE.CMD      GETVAX.CMD     PRIVAX.CMD     FILES.CMD
DUMPDISK.CMD   FORMAT.CMD     LOAD.CMD       COPYF.CMD      CREATE.CMD
FILETEST.CMD   FREE.CMD       RUN.CMD        BASIC.CMD      MTESTWY.CMD
EPROM.CMD      WPS.CMD        DELETE.HLP     BUILD.HLP      DATE.HLP
APPEND.HLP     ASN.HLP        FREE.HLP       GET.HLP        VERIFY.HLP
JUMP.HLP       CAT.HLP        COPYF.HLP      LIST.HLP       EXEC.HLP
COPY.HLP       I.HLP          P.HLP          PRINT.HLP      O.HLP
XOUT.HLP       RENAME.HLP     SAVE.HLP       QCHECK.HLP     WHELP.TXT
GUIDE.TXT      MEMO.TXT       MONCALLS.TXT   SINETAB.TXT    STARTUP.TXT
STARTUP2.TXT   DIVIDE.TXT     D_MULT.TXT     MOUSE.TXT      WELCOME.TXT
COPYD.TXT      SI.TXT         MOUSE.BIN      MOUSE.SYM      SI2.TXT
KERNEL.TXT
```

FIG. 3.

RESULT OF "CAT"



```
CATALOG OF DRIVE NUMBER 0
DISK: FLTS1085  #1

 NAME    TYPE  SIZE  PRT

ERRORS  .SYS     9
FLEX    .SYS    23
CAT     .CMD     3
DIR     .CMD     5
COPY    .CMD     5
LIST    .CMD     3
ASN     .CMD     1
DELETE  .CMD     2
RENAME  .CMD     1
TTYSET  .CMD     2
P       .CMD     1
SAVE    .CMD     2
EDIT    .CMD    28
ASMB    .CMD    48
ASM     .CMD    52
APPEND  .CMD     3
BUILD   .CMD     1
EXEC    .CMD     1
JUMP    .CMD     1
DATE    .CMD     2
         |
         |
         |
         |
         |
         |
         |
MOUSE   .TXT    15
WELCOME .TXT     2
COPYD   .TXT     1
SI      .TXT     1
MOUSE   .BIN     2
MOUSE   .SYM     2
SI2     .TXT     1
KERNEL  .TXT    34

SECTORS LEFT = 708
```

FIG. 4.

RESULT OF "DIR"

```
DIRECTORY OF DRIVE NUMBER 0
DISK: FLTS1085  #1    CREATED: 10-SEP-85

FILE#   NAME    TYPE  R  BEGIN   END    SIZE    DATE      PRT


    1  ERRORS  .SYS  R  01-01  01-09     9  10-SEP-85
    2  FLEX    .SYS     01-0A  01-20    23  10-SEP-85
    3  CAT     .CMD     02-01  02-03     3  10-SEP-85
    4  DIR     .CMD     02-04  02-08     5  10-SEP-85
    5  COPY    .CMD     02-09  02-0D     5  10-SEP-85
    6  LIST    .CMD     02-0E  02-10     3  10-SEP-85
    7  ASN     .CMD     02-11  02-11     1  10-SEP-85
    8  DELETE  .CMD     02-12  02-13     2  10-SEP-85
    9  RENAME  .CMD     02-14  02-14     1  10-SEP-85
   10  TTYSET  .CMD     02-15  02-16     2  10-SEP-85
   11  P       .CMD     02-17  02-17     1  10-SEP-85
   12  SAVE    .CMD     02-18  02-19     2  10-SEP-85
   13  EDIT    .CMD     02-1A  03-15    28  10-SEP-85
   14  ASMB    .CMD     03-16  05-05    48  10-SEP-85
   15  ASM     .CMD     05-06  06-19    52  10-SEP-85
   16  APPEND  .CMD     06-1A  06-1C     3  10-SEP-85
   17  BUILD   .CMD     06-1D  06-1D     1  10-SEP-85
   18  EXEC    .CMD     06-1E  06-1E     1  10-SEP-85
   19  JUMP    .CMD     06-1F  06-1F     1  10-SEP-85
   20  DATE    .CMD     06-20  07-01     2  10-SEP-85
   21  O       .CMD     07-02  07-03     2  10-SEP-85
   22  LINK    .CMD     07-04  07-04     1  10-SEP-85
                              |
                              |
                              |
                              |
                              |
   72  MEMO     .TXT    0D-18  0F-10    57  10-SEP-85
   73  MONCALLS.TXT     0F-11  0F-17     7  10-SEP-85
   74  SINETAB .TXT     0F-18  0F-1B     4  10-SEP-85
   75  STARTUP .TXT     0F-1C  0F-1C     1  10-SEP-85
   76  STARTUP2.TXT     0F-1D  0F-1D     1  10-SEP-85
   77  DIVIDE  .TXT     0F-1E  0F-20     3  10-SEP-85
   78  D_MULT  .TXT     10-01  10-02     2  10-SEP-85
   79  MOUSE   .TXT     10-03  10-11    15  10-SEP-85
   80  WELCOME .TXT     10-12  10-13     2  10-SEP-85
   81  COPYD   .TXT     10-1A  10-1A     1  17-SEP-85
   82  SI      .TXT     10-1B  10-1B     1  17-SEP-85
   83  MOUSE   .BIN     10-16  10-17     2  10-SEP-85
   84  MOUSE   .SYM     10-18  10-19     2  10-SEP-85
```

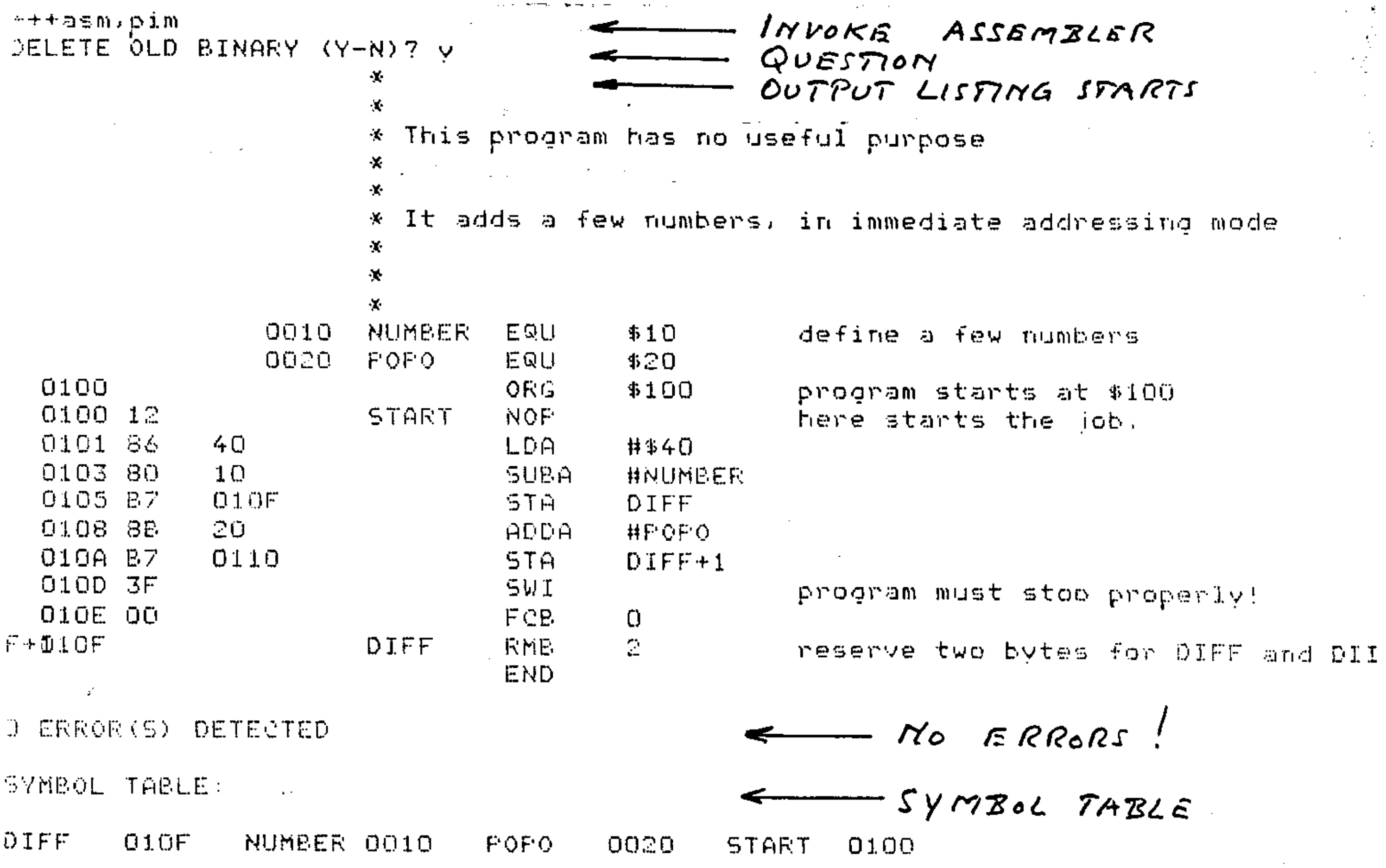
FIG. 5.

```
+++edit,pim                          ←  EDIT COMMAND
DELETE BACKUP FILE (Y-N)? y          ←  EDITOR ASKS QUESTION
#^p!                                 ←  LIST WHOLE FILE
    1.00=NUMBER EQU $10
    2.00=POPO EQU $20
    3.00= ORG $100
    4.00=START NOP
    5.00= LDA #$40
    6.00= SUBA #NUMBER
    7.00= STA DIFF
    8.00= ADDA #POPO
    9.00= STA DIFF+1
   10.00= SWI
   11.00= FCB 0
   12.00=DIFF RMB 2
   13.00= END
#0i                                  ←  INSERT BEFORE LINE 1
     .10= TTL FASULO                 ←  EDITOR TYPES .10=
     .20=*                              USER TYPES THE LINE.
     .30=*
     .40=* This program has no useful purpose
     .50=*
     .60=*
     .70=* It adds a few numbers, in immediate addressing mde
     .80=* *
     .90=*
    1.00=*
    1.10=#                           ←  TYPING # STOPS THE EDITOR
SOME LINES RENUMBERED                   FROM ASKING MORE LINES.
    1.00=*
#renumber                            ←  ANOTHER EDITOR COMMAND
#^p!                                 ←  LIST ENTIRE FILE AGAIN
    1.00= TTL FASULO
    2.00=*
    3.00=*
    4.00=* This program has no useful purpose
    5.00=*                                             COMMENTS ADDED
    6.00=*
    7.00=* It adds a few numbers, in immediate addressing mde
    8.00=*
    9.00=*
   10.00=*
   11.00=NUMBER EQU $10
   12.00=POPO EQU $20
   13.00= ORG $100
   14.00=START NOP
   15.00= LDA #$40
   16.00= SUBA #NUMBER
   17.00= STA DIFF
   18.00= ADDA #POPO
   19.00= STA DIFF+1
   20.00= SWI
   21.00= FCB 0
   22.00=DIFF RMB 2
   23.00= END
#7c/mde/mode/                        ←  MAKE CORRECTIONS
    7.00=* It adds a few numbers, in immediate addressing mode
#11/c/     / define a few numbers
?                                    ←  EDITOR DOES NOT UNDERSTAND
#11c/      / define a few numbers    ←  AGAIN
?
#11c/$10/$10 define a few numbers/   ←  NOW IT UNDERSTOOD
   11.00=NUMBER EQU $10 define a few numbers
```

FIG.-6.

```
#13o                                              <------- MODIFY LINE 13 WITH "OVERLA
   13.00= ORG $100                                <------- ORIGINAL
 OVERLAY                program starts at $100  <----- OVERLAY
   13.00= ORG $100  program starts at $100      <----- RESULT
#14c/$40/$40 here we sta  - rt our job/
?                                               <----- EDITOR DOES NOT
#14p                                                   UNDERSTAND.
   14.00=START NOP                   <----- PROVIZ(STRING $40 DOES NOT EXIST IN LINE 1
#14c/-  /NOP/NOP here starts the job?
   14.00=START NOP here starts the job? <----- NOW IT IS ok
#14o
   14.00=START NOP here starts the job? <----- GET RID OF ?
 OVERLAY
   14.00=START NOP here starts the job.
#p!
   14.00=START NOP here starts the job.       <------ LIST FROM CURRENT LINE
   15.00= LDA #$40                                              TO END
   16.00= SUBA #NUMBER
   17.00= STA DIFF
   18.00= ADDA #POPO
   19.00= STA DIFF+1
   20.00= SWI
   21.00= FCB 0
   22.00=DIFF RMB 2
   23.00= END
#20c/SWI/SWI program must stop properly!/ <--- ADD MORE COMMENTS
   20.00= SWI program must stop properly!      USING C AND O
#22o                                       <-----
   22.00=DIFF RMB 2
 OVERLAY                reserve two bytes f  or  .DIFF and DIIF+1
   22.00=DIFF RMB 2  reserve two bytes for DIFF and DIIF+1
#^p!                                               FINAL LISTING
    1.00= TTL FASULO
    2.00=*
    3.00=*
    4.00=* This program has no useful purpose
    5.00=*
    6.00=*
    7.00=* It adds a few numbers, in immediate addressing mode
    8.00=*
    9.00=*
   10.00=*
   11.00=NUMBER EQU $10 define a few numbers
   12.00=POPO EQU $20
   13.00= ORG $100  program starts at $100
   14.00=START NOP here starts the job.
   15.00= LDA #$40
   16.00= SUBA #NUMBER
   17.00= STA DIFF
   18.00= ADDA #POPO
   19.00= STA DIFF+1
   20.00= SWI program must stop properly!
   21.00= FCB 0
   22.00=DIFF RMB 2  reserve two bytes for DIFF and DIIF+1
   23.00= END
#S                                          <------- QUIT THE EDITOR
```

FIG. 7.



```
*++asm,pim
DELETE OLD BINARY (Y-N)? y
                         *
                         *
                         * This program has no useful purpose
                         *
                         *
                         * It adds a few numbers, in immediate addressing mode
                         *
                         *
                         *
                  0010   NUMBER   EQU     $10         define a few numbers
                  0020   POPO     EQU     $20
  0100                            ORG     $100        program starts at $100
  0100 12                START    NOP                 here starts the job.
  0101 86    40                   LDA     #$40
  0103 80    10                   SUBA    #NUMBER
  0105 B7    010F                 STA     DIFF
  0108 8B    20                   ADDA    #POPO
  010A B7    0110                 STA     DIFF+1
  010D 3F                         SWI                 program must stop properly!
  010E 00                         FCB     0
F+010F                   DIFF     RMB     2           reserve two bytes for DIFF and DII
                                  END

0 ERROR(S) DETECTED

SYMBOL TABLE:

DIFF    010F    NUMBER 0010    POPO    0020    START   0100
```

FIG. 8.

```
+++ LOAD, PIM.BIN
+++ MON                                        ←——— FLEX COMMANDS "LOAD" AND "MON
ROSY  >                                        ←——— BACK IN ROSY
ROSY  >decode 100 110                          ←——— CHECK PROGRAM IS IN MEMORY
0100    12            NOP
0101    86    40      LDA     #$40
0103    80    10      SUBA    #$10
0105    B7    010F    STA     $010F
0108    8B    20      ADDA    #$20
010A    B7    0110    STA     $0110
010D    3F    00      MON     #$00
010F    30    50
ROSY  >m-li 100 11F
                                                   MEM-LIST
ADDR    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100   12 86 40 80 10 B7 01 0F 8B 20 B7 01 10 3F 00 30    ..@..........?.0
0110   50 C8 A5 EC 4A 44 56 44 56 44 56 17 1F F2 ED 4A    P...JDVDVDV....J
ROSY  >m-mo 10F
010F 30 00                                  ←——— SET TWO BYTES TO ø.
0110 50 00
0111 C8 .
ROSY  >
ROSY  >go    100                               RUN THE PROGRAM

ROSY  >m-li 100 11F
                                                  RESULTS ARE HERE!
ADDR    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100   12 86 40 80 10 B7 01 0F 8B 20 B7 01 10 3F 00 30    ..@..........?.0
0110   50 C8 A5 EC 4A 44 56 44 56 44 56 17 1F F2 ED 4A    P...JDVDVDV....J
ROSY  >f
                                       BACK TO FLEX
+++delete,pim.bin                      DELETE A FILE

DELETE "0.PIM.BIN" ? y                 IS THIS WHAT YOU WANTED?
ARE YOU SURE? y
+++files,0                            CHECK THAT FILE HAS BEEN DELETED.


FILES ON DRIVE NUMBER 0

ERRORS.SYS      FLEX.SYS        CAT.CMD         DIR.CMD         COPY.CMD
LIST.CMD        ASN.CMD         DELETE.CMD      RENAME.CMD      TTYSET.CMD
P.CMD           SAVE.CMD        EDIT.CMD        ASMB.CMD        ASM.CMD
APPEND.CMD      BUILD.CMD       EXEC.CMD        JUMP.CMD        DATE.CMD
O.CMD           LINK.CMD        VERSION.CMD     PROT.CMD        VERIFY.CMD
PRINT.CMD       QCHECK.CMD      I.CMD           XOUT.CMD        PUTLDR.CMD
HELP.CMD        TMODE.CMD       GETVAX.CMD      PRIVAX.CMD      FILES.CMD
DUMPDISK.CMD    FORMAT.CMD      LOAD.CMD        COPYF.CMD       CREATE.CMD
FILETEST.CMD    FREE.CMD        RUN.CMD         BASIC.CMD       MTEST09.CMD
EPROM.CMD       WPS.CMD         DELETE.HLP      BUILD.HLP       DATE.HLP
APPEND.HLP      ASN.HLP         FREE.HLP        GET.HLP         VERIFY.HLP
JUMP.HLP        CAT.HLP         COPYF.HLP       LIST.HLP        EXEC.HLP
COPY.HLP        I.HLP           P.HLP           PRINT.HLP       O.HLP
XOUT.HLP        RENAME.HLP      SAVE.HLP        QCHECK.HLP      WHELP.TXT
GUIDE.TXT       MEMO.TXT        MONCALLS.TXT    SINETAB.TXT     STARTUP.TXT
STARTUP2.TXT    DIVIDE.TXT      D_MULT.TXT      MOUSE.TXT       WELCOME.TXT
COPYD.TXT       SI.TXT          MOUSE.BIN       MOUSE.SYM       SI2.TXT
KERNEL.TXT      PIM.TXT         PIM.BAK
                                                PIM.BIN HAS GONE.
  [illegible] ,pim.txt,prime.txt                                        FIG. 9
```