DECUS
PROGRAM LIBRARY

DECUS NO.	8-466B
TITLE	RL MONITOR SUBSYSTEMS P?S-08-1.1B
AUTHOR	Richard Lary Submitted by: Stanley Rabinowitz
COMPANY	Polytechnic Question Society Brooklyn, New York
DATE	March 1, 1971
SOURCE LANGUAGE	PAL III

DECUS Program Library Write-up

DECUS NO. 8-466B

The RL-Binary Loader (BIN) is a system on all tapes containing the RL Monitor (WCFMPG version). It is used to load (and execute) programs stored as binary files on the tape. These programs originally got on the tape as binary files by running PAL with the appropriate parameter and then doing a BSAVE.

The general form of the command to use BIN is as follows:

RUN BIN,$file_1$,$file_2$,...,$file_n$=param

where $file_1$,...$file_n$ are the names of the binary files to be loaded into core. Not more than 5 binary files can be simultaneously loaded into core in this manner, i.e., n can not exceed 5.

param is an octal number representing the starting address of the program. param may be any octal constant from 1 to 7777. param may not be Ø, that is, the starting address of your program may not be Ø.

If param is specified, after the program has been loaded into core, control is transfered to the location specified by param, thus automatically starting your program.

If param is Ø or omitted entirely, the program gets loaded into core and then the computer halts. This halt will occur at location 7773, with the AC cleared, and a 76Ø2 in the memory buffer. You may then manually start your program by hitting load address (with the starting address set on the switch register) and then depressing start.

Note that a manual start may be different than an automatic start, because when the start key is depressed, other effects occur besides starting, such as the AC and Link being cleared, and all flags (from I/O devices) being cleared.

If $file_1$ is the symbol, $, it represents the binary file located in the BIN, (blocks 22-57 on the tape.) If any other filename is a $, it represents the user's file named $ (located in blocks 4Ø-57 on the tape).

If more than 5 files are specified, any after the fifth are ignored. If no files at all are specified, i.e., only the command RUN BIN is given (possibly followed by a starting address parameter), the binary file located in the BIN is loaded.

The command RUN BIN is identical with RUN BIN,$.

Since the binary loader resides in the last page of core (76ØØ-7777), no data must be loaded into this area without destroying the loader. Thus, your source program should not contain any statements or data which directly loads into this area in core.

The RL Binary Loader is different from the DEC Binary loader, since it loads binary programs from tape. These programs are stored using the special RL-Binary Format which is different from the standard DEC Binary Format. The RL-Binary Loader cannot be used to load paper tapes punched in RIM or BIN format.

Examples: The usual sequence of commands using the RL-Binary Loader are as follows:

```
RUN PAL=13,MYPROG,PROG2
RUN BIN=2ØØ
```

```
RUN PAL=13,MYPROG
BSAVE BPROG
RUN PAL=13,MYP2
BSAVE B2
RUN BIN,BPROG,B2=34ØØ
```

The System EDIT

EDIT is a system which is used in conjunction with the RL Monitor and editor. It is used to allow commands to the RL Monitor (editor) to come from a file on DECTAP instead of from the teletype.

To use EDIT, one or more RL source files are prepared containing commands to the editor, such as LOAD, SAVE, RESEQUENCE, etc. These are saved as source files on the tape as usual. Then EDIT is run specifying these files; thus causing all the commands on the tape in these files to be executed. After all commands have been executed, further commands can then be entered as usual through the teletype.

The general form of the command to EDIT is as follows:

RUN EDIT,$file_1$,$file_2$,...,$file_n$

where $file_1$,...,$file_n$ are the names of the files containing the commands to be passed to the editor. EDIT uses no parameters.

The number of files which can be passed to EDIT is from 1 to 15. If n is greater than 15, the monitor will bomb.

Although the files were created with line numbers, the line numbers are not passed to EDIT, just as the line numbers are not passed to other systems such as PAL. Thus if you want to use EDIT to input source text into a file on dectape from a different file on dectape, you must first give a sequence command to the editor. If this is the last command to the editor (and only source lines follow), control will return to accepting commands from the teletype and you can get out of sequence mode by hittind EOT (control D) on the teletype. If you wish to get out of sequence mode automatically and continue accepting editor commands from the tape file, the special character, \ (back slash), accomplishes this and should be the last character of the source lines. It must be immediately followed by a monitor command with no intervening carriage returns or spaces. Back slash does not affect the echo flag. (EOT always turns echo on.) It is convenient to keep echo off while moving or merging files in this manner since no time is wasted waiting for the teletype to be ready.

You can not give a rewind command while in EDIT mode. The EDIT program will lose control if a command is executed which rebootstraps or does not return control to the editor, such as MONITOR or any RUN command. Thus if a RUN command must be given (such as a RUN PAL or RUN BIN) it should be the last command in the edit file.

The edit file is often formed in the $ file, because this file is at the beginning of the tape. EDIT reads in a block at a time of commands from the edit file, so if your edit file is large, you must be careful not to switch tapes while in the middle of an EDIT (unless you are very careful and make sure you have the same edit source file in the same position on both tapes).

Care must be taken not to change the edit source file during an EDIT.

The System EDIT

EDIT is a versatile system which is found on all RL tapes. It resides in two blocks on the tape and when loaded into core it resides in page 76ØØ. It can be used to perform a myriad of activities which require execution of a large sequence of operator commands to the monitor without the operator having to continually enter these commands on-line. Some useful examples follow. The user will be able to devise many other important uses of EDIT.

1. To make long listings (while you go out to eat):

```
SCRATCH
10  scratch
20  LOAD FILEA
30  LIST
40  LOAD FILEB
50  LIST
:   :
:   :

SAVE $
RUN EDIT,$
```

2. To merge two files, FILEA and FILEB in the sense that you will create a new file called FILEC which will contain all the statements from FILEA followed by all the statements from FILEB.

```
SCRATCH
10  ECHO
20  SCRATCH
30  SEQUENCE 1ØØ 1Ø
SAVE $
SCRATCH
10  \ECHO
20  SAVE FILEC
SAVE ED2
RUN EDIT,$,FILEA,FILEB,ED2
```

3. To clean up a tape by rearranging the files:

```
SCRATCH
10  LOAD JOE
20  DELETE JOE
30  SAVE FILE1
40  LOAD TEST
50  DELETE TEST
60  SAVE FILE2
:   :
:   :
80  CATALOG
SAVE EDDY
RUN EDIT,EDDY
```

4. To save typing time for certain commands:

```
SCRATCH
10 RUN PAL=4013,SWØ,SW1,SW2,SW3,SW4,SW5,SW6,SW7,SW8,SW9
SAVE $
```

5. To transfer user files from one tape to another.

```
SCRATCH
10 LOAD A
20  HALT
30  SYS
40  SAVE A
50  HALT
60  SYS
70  LOAD B
80  HALT
90  SYS
100 SAVE B
110 HALT
120 SYS
130 LOAD C
:   :
:   :
200 SAVE Z
210 CATALOG

SAVE $
(change tapes)
SAVE $
(remount original tape)
RUN EDIT,$
(Thereafter, each time the machine halts, you switch
 tapes and hit continue, keeping write enabled only
 for the new tape. This process continues until the
 new catalog on the new tape is listed.)
```

Note that it was important to save the edit file on both tapes.

Notes: If EDIT detects an illegal command, the error message, WHAT?, is printed as usual and command execution continues with the next line in the edit file. In particular, if while doing a merge, the workspace gets filled up, WHAT? messages will begin to flow. All error messages and monitor statements print even if the ECHO switch is off. The only way to prematurely and abruptly terminate an EDIT is to hit stop and rebootstrap.