DECUS
PROGRAM LIBRARY

DECUS NO.	8-389
TITLE	MINI-MONITOR, A SECONDARY DISK MONITOR FOR THE PDP-8
AUTHOR	R. S. Lewis
COMPANY	Submitted by: John C. Kerr Science Research Council, Rutherford Laboratory Chilton, Didcot, Berkshire, England
DATE	August 1970
SOURCE LANGUAGE	PAL D

MINI-MONITOR, A SECONDARY DISK MONITOR FOR THE PDP-8

DECUS Program Library Write-up DECUS NO. 8-389

Abstract

A system is described which allows a large number of program core images to be saved on disk by name, but independently of the DEC Monitor System. The restrictions imposed by the limit on the number of file-names normally available are thus avoided.

Introduction

The program was written for a multi-disk PDP-8 connected on-line to one of the external beam-lines of a particle accelerator. Before Mini-Monitor was written, a number of programs were in operation for the control of magnets, collimators etc., in the beam-line, and it was evident that the decreasing number of unused file-names (a total of 63_{10} are available) under which to store programs was about to become an embarrassment as far as any further development in user software was concerned. This was a clear case of "Necessity is the mother of invention", and so Mini-Monitor was born. (It would, of course, be possible to expand the DEC Monitor, but a large amount of work would be involved in modifying all system programs in order to make them compatible with the new system).

Organisation

Briefly, the organisation of the program is as follows:-

A master ("system") file is created on disk under the DEC Monitor which is large enough to contain the Mini-Monitor program itself and sufficient space in which to store as many program core images as is desired. Programs are then saved within this file by Mini-Monitor, which keeps a record of file-names, core specifications and block allocations in DN and SAM blocks very similar to those used in the DEC Monitor. Provision is made in the structure of the DN blocks for a maximum of 125_{10} file-names, and in the SAM blocks for a master file of up to 1000_8 blocks (or approximately 2 disks) in length. The master file can very easily be expanded by increasing the number of SAM blocks, and with a little more trouble the number of available file-names can also be increased.

In the system for which the program is written, the master-file is allocated the name "MINI" under the DEC Monitor, and runs from block 600 to the end of disk 2. The program itself occupies 24_8 blocks of the file (600 - 623), including DN and SAM blocks and 3 scratch blocks. When the program is running, it occupies a maximum of 3 contiguous pages of core at any one time (6200 - 6777). Providing that the symbolic tape is reassembled with appropriate origin settings, Mini-Monitor can be run anywhere else in Field 0, except the top page of core, since the program uses the DEC Sysio Head for disk/core transfers. The starting block of the master file on disk may also be changed, providing the constant FSTBLK is redefined on the symbolic tape.

Operation

The program is entered by calling it from disk in the normal way:-

```
.MINI⤶
#
```

It responds with a number sign (#) typeout. The program is now in command mode and awaits user input. There are four legal commands that may be given; they are described below.

SAVE

The SAVE command is used in virtually the same way as its DEC equivalent. There are, however, a number of important differences:-

a) No distinction is made in Mini-Monitor between "system" and "user" programs (DEC nomenclature), and the character immediately following the file-name in a SAVE command is <u>always</u> a colon.

b) A single octal number given as a core specification is illegal. Lower and upper core limits separated by a dash <u>must</u> be specified.

c) Only contiguous saves are possible.

d) Programs cannot be saved from the three pages of core used by Mini-Monitor.

Examples:-

```
#SAVE PRG1:200-1577;200⤶

#SAVE PRG2:3200-14577;4000⤶          (SAVE from Field 1)
```

CALL

This is used in exactly the same way as the DEC Monitor CALL:-

```
#CALL PRG1⤶
```

If the entry point specified in the DN entry for the file PRG1 is zero, control returns to Mini-Monitor on completion of the CALL. If non-zero, control is transferred to the address specified by the entry point, in the Field from which the program was saved.

RUBB

The RUBB command deletes a file from the directory of Mini-Monitor:-

```
#RUBB PRG1⤶
```

After execution of the above command, the DN entry and all the SAM entries for the file PRG1 will have been set to zero.

LIST

The LIST command causes Mini-Monitor to output on the Teletype a list of the files in the directory, preceded by the number of free blocks left in the master file. For example:-

```
#LIST⤶

FB=723

  1.RS L
  2.PRG1
  3.PRG2
```

In the above example, files 1-3 are the only occupied ones, file number 1 being Mini-Monitor itself. The name (RS L) has an imbedded blank to protect it from accidental deletion. (The letters are the author's initials!). Any user input during execution of the LIST command terminates output and causes a return to command mode.

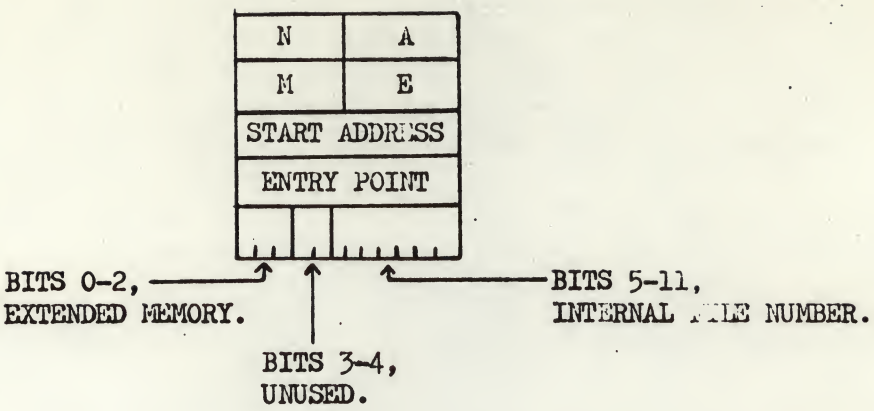
Special Keys

RUBOUT causes the present command string to be ignored, and initiates a Monitor restart.

CTRL/C causes a jump to 7600, thus returning control to the DEC Monitor.

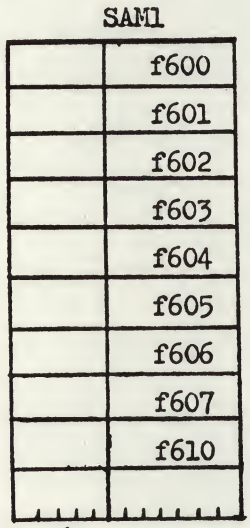
DN and SAM Block Formats

The format of entries in the DN blocks is as follows:-



File-name characters are packed 2 to a word in 6-bit stripped Ascii + 40. There are 5 DN blocks, 25_{10} entries per block, and the last 3 words of each block are unused. The blocks are link chained in the normal way, the link of DN5 being zero.

The SAM blocks are structured as follows:-



fnnn = internal file number of file occupying block nnn. A zero word indicates that the block is unoccupied.

Each SAM block (there are 4) contains a record of 200_8 blocks of the master file. They are link chained in the same manner as the DN blocks.

When the Mini-Monitor system is initially created on disk, the first DN entry is for Mini-Monitor itself, and the first 24_8 words of SAM1 are set to 001 (blocks 600-623 are occupied by Mini-Monitor).

Mini-Monitor Block Allocation

Block Number	Link	Contents	Core Page
600	000	Command Decoder	6600
601	000	Rubb	5000
602	607	Page 1 of List	5400
603	000	Call	5600
604	605	Page 1 of Save	6400
605	606	Page 2 of Save	6200
606	000	Page 3 of Save	6000
607	000	Page 2 of List	5200
610	611	Scratch	-
611	612	Scratch	-
612	000	Scratch	-
613	614	DN1	-
614	615	DN2	-
615	616	DN3	-
616	617	DN4	-
617	000	DN5	-
620	621	SAM1	-
621	622	SAM2	-
622	623	SAM3	-
623	000	SAM4	-

The column headed "Core Page" indicates the page from which each part of Mini-Monitor must be written on to disk when the object tape has been loaded into core. When the program is running, however, locations 5000-6177 are unused.

Creation of the System on Disk

Assuming, as before, that the program is intended to run from 6200-6777 and that the starting block of the master file is to be block 600, the following procedure has been found relatively straightforward.

a) By the deletion and manipulation of existing DEC files, arrange for the first free block on disk to be block 600, and for the area intended for the master file to be unused by the DEC Monitor.

b) Load the object tape of Mini-Monitor into core, and save the Command Decoder as follows:-

```
.SAVE MINI!6600-6777;6601↵
```

This procedure creates the initial file, beginning at block 600.

c) Using a program such as DISKLOOK (Decus No. 8-111) the next step is to expand the master file to the required length by alteration of the DEC SAM blocks. Each SAM entry must be filled with the internal file number of the file MINI. The entry for block 600 will give the number.

d) Now the master file has been created, the rest of Mini-Monitor (which should still be in core) can be written on to disk.

To do this, obtain a binary tape of the following (or similar) short routine, and load into core.

```
Set S.R. = 200;              LOAD ADD
Set S.R. = Block Number;     START
Set S.R. = Core Address;     CONT
Set S.R. = Link Number;      CONT
```

Repeat this process for each of the Mini-Monitor program blocks, according to the block allocation table (above).

```
        *200
        LAS                         /GET BLOCK NUMBER
        DCA BLOCK
        HLT
        LAS                         /GET CORE ADDRESS
        DCA CORE
        HLT
        LAS                         /GET LINK NUMBER
        DCA LINK
        JMS I SYSIO                 /START THE TRANSFER
        5                           /WRITE FUNCTION
BLOCK,  0
CORE,   0
LINK,   0
        HLT                         /ERROR RETURN, AC=0
        CMA
        HLT                         /AC = 7777 IF NO ERROR
SYSIO,  7642
```

The DN, SAM and scratch blocks can be written on to disk using the same routine, from a core page that has been set to zero in every location.

e) Using DISKLOOK again, set the first 24_8 words of SAM1 to 001. The first 5 words of DN1 can then be set to an appropriate entry for the Mini-Monitor program. Start address, entry point and Field numbers are unimportant, since the program should never be called from within itself, but the internal file number must be 001. It is advisable to include an imbedded blank in the file-name, since the program can not then be deleted from the master file.

f) Return to the DEC Monitor, call MINI into core, and see if it works!

Restrictions

Mini-Monitor does no swapping-out, therefore programs cannot be saved from the 3 pages used by it.

Only contiguous saves are possible.

R.S. Lewis