DECUS
PROGRAM LIBRARY

DECUS NO.	8-336
TITLE	DECTAPE LIBRARY SYSTEM MODIFICATIONS
AUTHOR	William Leal
COMPANY	Submitted by: Ernest Hayden Speech Communications Research Laboratory Santa Barbara, California
DATE	December 4, 1970
SOURCE LANGUAGE	PAL III

INTRODUCTION to DECtape Library System modifications

The tapes and documents provided permit the PDP-8 user running under the DEC-Ø8-SUA1-LA DECtape Library System to call in one program from another, and the ability to write programs which may be executed, and, when completed, leave storage undisturbed, much like the system programs. These programs are called phantom programs. Users running under DIGITAL-8-7-S Rev 7/25/66 may make use of the phantom features as is, but MODIND and OCTDMP will have to be modified to accomodate the different directory format.

Also included is IOFMAG to help the user understand the function of MODIOF.

MODIND requires a simple hardware modification which is described on an accompanying logic diagram.

As an overview:

IOFMAG is 4 routines which access TC-Ø1 DECTapes without the use of the interrupt.

MODIOF will modify a system tape so that IOFMAG is resident above the bootstrap. This permits the use of PHANTOM PROGRAMS.

MODIND will modify a system tape so that DYNAMIC OVERLAYS are possible if the specified hardware modification has been made.

LISTX is a phantom which will list the directory of a system tape showing not only the name of each program, but also where on the tape it is stored, the starting address, and the storage locations occupied when in core.

OCTDMP is a phantom which will print (in octal) specified storage locations.

BINPCH is a phantom which will punch in BIN loader format specified storage locations.

BINLDR is a phantom which will load into storage a tape in BIN loader format.

LISTX, OCTDMP, BINPCH & BINLDR are all on the DECtape provided. To add them to the user's system:

1) Mount the tape provided. Request LISTX. It will halt.
2) Dismount the tape provided. Mount the user's tape. Start at 7600. Update with parameters shown in LISTX documentation.
3) Repeat 1 & 2 for the other programs.

Note on the ASCII tapes provided: the assembler in use by the author has the psuedo OP's NPRINT, YPRINT, and EJECT to control pass 3 output. if the reciepient's assembler does not have these, they should be removed from the tapes provided.

Four subroutines are provided which access DECtapes on TC-Ø1 drives. In contrast to the routines provided by DEC, these routines do not use the interrupt. If the user does not require the use of the interrupt while doing DECtape I/O, the use of these routines has several advantages:

1) They occupy less storage (1Ø6 [1Ø] vs. 128 [1Ø] words).
2) The user need not establish page Ø linkages to service the interrupt.
3) Other devices (such as the teletype) will not interrupt the DECtape I/O. When, for instance, transferring from paper tape to DECtape, one need not buffer those characters which would have been read during DECtape output.

The subroutine linkages are as follows:

```
JMS DTREA/read, ac must = Ø
xxxx      /lst transfer location
uØØØ      /u = unit #
-n        /n = # of blocks to be transferred
b         /b = lst block to be transferred
xxxx      /instruction executed when read completed

JMS DTWRI/write, ac must = Ø
xxxx      /lst transfer location
uØØØ      /u = unit #
-n        /n = # of blocks to be transferred
b         /b = lst block to be transferred
xxxx      /instruction executed when write completed

JMS DTSAS/search and stop, ac must = block # wanted
uØØØ      /u = unit #
xxxx      /instruction executed when search and stop completed

JMS DTSNS/search and don't stop, ac must = Ø
     DTBLOC must be put = block # wanted
     DTUNIT must be put = uØØØ
     note:  the other subs will set DTBLOC & DTUNIT.
```

If a tape error occurs, the sub will go to the address specified in LDTERR which is initially 76ØØ.

The block length is specified in DT76ØØ, which is initially 76ØØ (-2ØØ).

The DTSNS routine differs from DTSAS only in a) linkages and b) the tape is in motion when the subroutine is completed.

MODIOF

This program will modify a system tape such that IOFMAG will be resident above the bootstrap when a user's program is in core. Thus the binary loader is no longer available on the modified tape. However, note BINLDR.

The resident IOFMAG entry points & pertinent locations are:

DTREA	is at	7626
DTWRI	"	7756
DTSAS	"	7673
DTSNS	"	7704
DTBLOC	"	7772
DTUNIT	"	7776
LDTERR	"	7777
DT7600	"	7770

To modify a system tape:

1) Mount system tape provided. Request MODIOF. It will halt after it has been loaded.
2) Dismount provided system tape. Mount tape to be modified.
3) Press start. The tape will be modified, and control will be passed to the system.

Note: It is suggested that the following 3 symbols be added to the assembler's symbol table:

```
DTREAD = 7626
DTWRIT = 7756
DTSRCH = 7673
```

PHANTOM PROGRAMS

A phantom program is a program which may be called and executed, and when normal exit is taken, restores positions 0-7577 to what they were before the phantom program was called. Thus the system programs are all phantoms--DELETE, UPDATE, etc.

Phantom programs may be written by users, too, if their system tape contains IOFMAG resident above the bootstrap. This is done by recognizing the fact that when their phantom program is called in, locations 6000-7577 of the program previously in core are written onto blocks 1-7 of the system tape. Thus the phantom program may occupy these locations. When exit is taken from the phantom, the routine described below is executed, which restores locations 6000-7577 of the previous program, and returns to the system. If ESCAPE is typed at this point, then 0-7577 will be as it was before the phantom was called in.

To restore 6000-7577, execute the following code when the phantom is ready to exit:

```
EXIT, TAD OP5200/JMP TO 7600 WHEN THROUGH READING
  DCA I OP7653
  JMS I LREAD   /RESTORE HIGH CORE
   6000
   0000
   -7
   1
OP7653, 7653
OP5200, 5200
LREAD, DTREAD
```

DYNAMIC OVERLAYS

Using this hardware/software modification, it is possible for a program to call in another program and execute it. This sort of procedure is called an overlay. Since it is done from a running program, it is called a dynamic overlay.

The hardware modification involved is to define a special flag for the system (see accompanying logic diagram). This flag can be set, cleared, and tested. A power clear (via the START switch) will clear it. It performs no other function. This flag may be tested by the system to see if an overlay is desired. Device code 33 was chosen and the following additions made to the symbol table:

```
skov = 6331/skip if overlay flag=1
stov = 6332/set overlay flag=1
clov = 6334/set overlay flag=0
```

Use of dynamic overlays is perhaps best explained by an illustration:

```
*0
abc-1

.............

abc, 301
 302
 303
 215

.............

exit, stov
  jmp i .+1
   7600
```

In this code, jumping to "exit" causes the overlay flag to be set. Control is then passed to the system which, sensing the overlay flag, clears it and gets from storage beginning at abc. The getting is done using the address in location 0, which is a pointer to the address-1 of the location of the name of the program. The name is echoed on the Teletype.

If the program requested is not present, a halt will be executed after a question mark is typed. The user has two alternatives at this point: he may restart at 7600 (in which case locations 6000 through 7577 of the calling program are lost), or he may

1) reset location 0
2) correct the name of the program
3) restart at 7001

The teletype flag is cleared initially by this new index program, but the teleprinter flag is left up when the called program is entered.

NOTE:
The name of the program to be called must be in the range of 0-5777 inclusive.

EXAMPLE OF OVERLAY:

```
*200/overlay - 1
  tad dover
  dca 0
  stov
  jmp i .+1
   7600
dover, .
  317/over2
  326
  305
  322
  262
  215
$


*200/overlay - 2
  tad dover
  dca 0
  stov
  jmp i .+1
   7600
dover, .
  317/over1
  326
  305
  322
  261
  215
$
```

These programs were updated as

```
program name  :overl
sa(octal)     :200
page locations:[200];
```

and

```
program name  :over2
sa(octal)     :200
page locations:[200];
```

These programs do nothing more than call each other dynamically: when overl is called, it sets location 0 to the location of the name of over2 minus one. It then sets the overlay flag and exits to the system. The system senses the overlay flag, and gets the program name from the locations specified by location 0. This causes over2 to be called in, which calls in overl, etc.

To modify a system tape:

1) Mount system tape provided. Request MODIND. It will halt after it has been loaded.
2) Dismount provided system tape. Mount tape to be modified.
3) Press START. The tape will be modified, and control will be passed to the system.

LISTX
start=6600
storage=[6600,7400]; (phantom program)

LISTX will print the directory of a system tape. On exit, storage is set to what it was before LISTX was called in. LISTX varies from INDEX in that LISTX prints not only the name of each program in the directory, but it also prints the block number at which it's stored, the starting address, and the core locations. It also prints the tape label, a feature to be added to the system later. This is printed as L=label.

flow:

read in directory
print label
print program data
restore high core
exit to system

OCTDMP
start=6000
storage=[6000-6200] (phantom)

OCTDMP will dump in octal specified storage locations on the Teletype It is a phantom program, so that it may be called while debugging a program, and, when execution is complete, the program being debugged will be restored to what it was before OCTDMP was called.

Flow

a, types BEGIN AT
  user replys with beginning storage location (octal), terminated by a non-numeric character
  if terminator = E, then go to c
  types END AT
  user replys with ending storage location (octal)
  if terminator not = M, go to b
  types MASK=
  user types in mask
b, specified locations are printed
  goes to a
c, restores 6000-7577 and exits to system

If a mask was specified, all values typed are ANDed with the mask.

Output is paginated.

Note: Block 10 of the system tape contains the bootstrap and resident IOFMAG.

Note: If any of locations 6000-7777 are requested for dumping, then they will be gotten off of blocks 1-10 of the system tape.

BINPCH
start=6000
storage=(6000) (phantom)

BINPCH will punch in BIN loader format specified storage locations on paper tape. It is a phantom program, so that it may be called in, and, when execution is complete, the original contents of storage will be restored to what they were before BINPCH was called.

Flow

a, halts
  user sets SR to # of blocks to be punched, presses continue
  if # of blocks to be punched = 0, then restores 6000-7577 and
    exits to system
b, halts
  user sets SR to starting address, presses continue
  halts
  user sets SR to ending address, presses continue
  punches block
  if more blocks, go to b
  punches check-sum
  punches trailer
  halts
  go to a

Note: Block 10 of the system tape contains the bootstrap and resident IOFMAG.

Note: If any of locations 6000-7777 are requested for dumping, then they will be gotten off of blocks 1-10 of the system tape.

BINLDR

start=6200

storage=[6000,6200] (phantom)

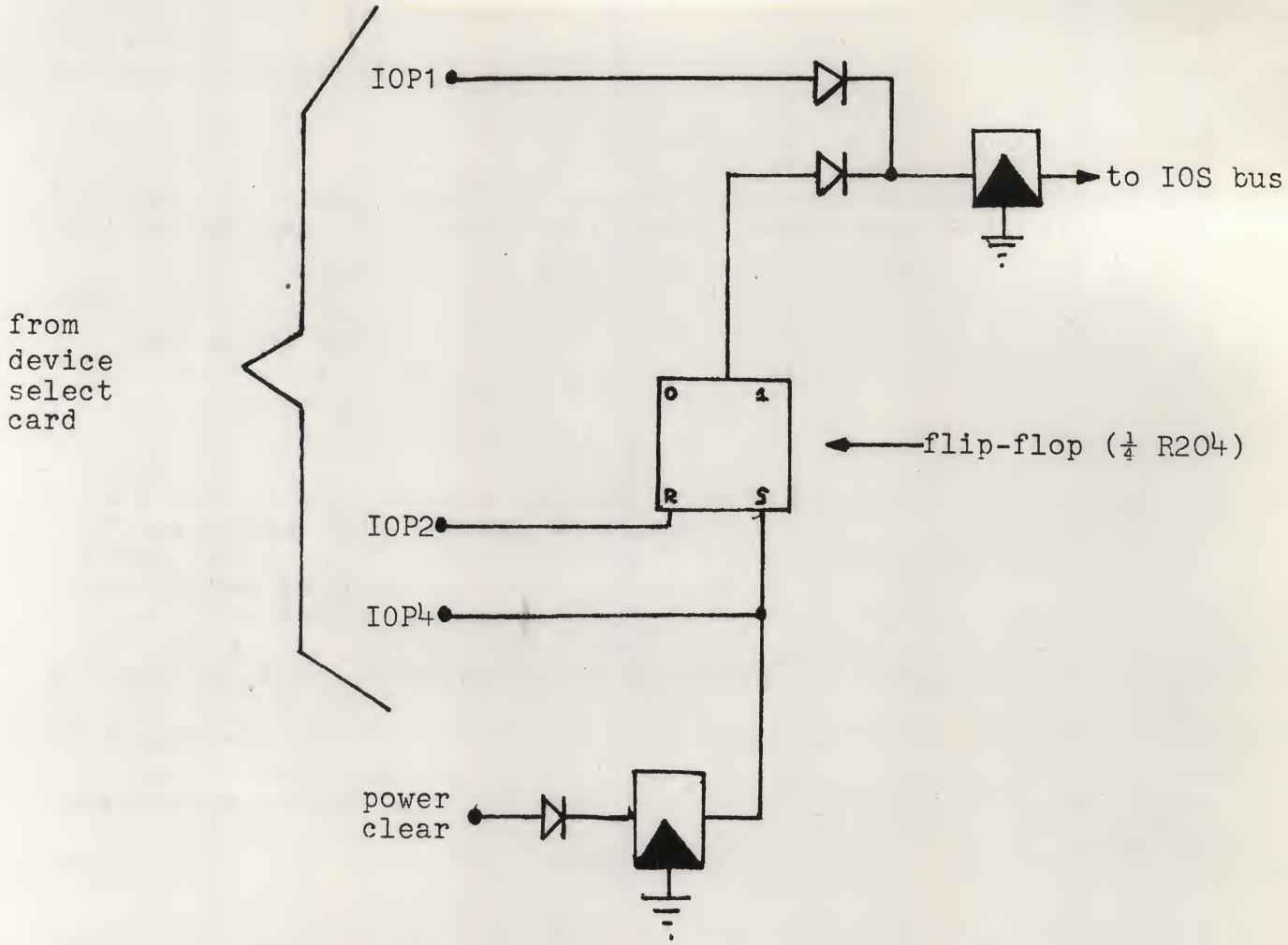
BINLDR will load into storage a tape in BIN loader format. It is a phantom program, so that it may be called in, and, when execution is complete, the original contents of storage, as modified by the tape read in, will be restored to what they were before BINLDR was called.

Flow

a, loads tape
  if check-sum = 0, restores 6000-7577 and exits to system
  displays check-sum and halts
  go to a

Note: If any of locations 6000-7577 are to be changed by the tape read in, they will be changed on blocks 1-7 of the system tape.

Note: If any of locations 7600-7777 are to be changed by the tape read in, this change will <u>not</u> be reflected after execution is complete.



IOS (skov)=IOP1
SET (stov)=IOP2
RESET(clov)=IOP4

OVERLAY FLAG LOGIC DIAGRAM