## IDENTIFICATION

Product Code:    DEC-08-SBAB-D

Product Name:    PDP-8 Disc System Builder

Date Created:    October 27, 1967

Maintainer:      Software Services Group

1       ABSTRACT

        This document describes a fast convenient keyboard oriented monitor which will enable the
user to efficiently control the flow of programs through his PDP-8.  The system is modular and open-
ended allowing the user to build the software components required in his environment.  The user may
specify the system device (disc or DECtape), the amount of core, number of disc units available and
the number, name and size of his resident system programs.

        This is a basic package to which DEC will add FORTRAN, Assembler, Editor, Peripheral
Interchange Program, DDT, and perhaps other programs, as they become available.  The system de-
scribed here includes all features of the monitor and allows the user full access to the disc as a program
storage medium.

        The System Monitor is designed to facilitate user/system communication.  By typing appro-
priate commands to the monitor the user may call the loader from the systems device, save a core image
on the systems device, or call a saved file into core from the systems device.

2       PRELIMINARY REQUIREMENTS

        PDP-8  (4K)
        3 cycle Data Break
        DF32 Disc (or TC01 DECtape)

3       SYSTEM GENERATION

3.1     Building the Monitor

        To load a monitor system on an empty disc or DECtape, perform the following sequence of
operations.

        a.   Load RIM Loader

        b.   Load Binary Loader

        c.   Load BUILD using the Binary Loader

        d.   Disable write protect on the desired device; DISC or DTA8

        e.   Start at $200_8$

        f.   Answer the questions (underlined) asked by the BUILD Program, as shown below,
to fit your hardware system.

                *TYPE SIZE OF CORE  (IN K)
                *4
                *HIGH SPEED PAPER TAPE?
                *YES

±DISC OR TAPE? ⟩
±DISC ⟩
±TYPE NUMBER OF DISC UNITS ⟩
±1 ⟩
± ⟩ (MAXIMUM 1 MINUTE DELAY)

± ⟩

g. You will get either:

1. ⟩　meaning the monitor and loader are correct or

2. WRITE ERROR.

If in a disc environment, you get the latter; start over at step a (you may have hardware problems). If you have DECtape, try a new DECtape at step d or try writing timing and mark tracks on the DECtape and start over. The builder will ask other questions as necessary allowing you to tailor your system to the hardware.

3.2　Building the System Library

a. Read the LOADER description (section 4.1) and CALL/SAVE description (section 4.2).

b. Type in appropriate commands as indicated in Loader description. By using the starting address of $7600_8$ ("ST = 7600"), control is returned to the monitor after load and each program may be saved. An example of loader use follows.

```
.LOAD ⟩
*IN-R: ⟩                    READ ONE TAPE FROM READER
↑                           LOADER FOUND DEVICE R: IS VALID
*OPT-2                      LOAD IN 2 PASS MODE (PROGRAM REQUIRES
                              USE OF CORE ABOVE 7000₈)
*ST = 7600 ⟩                JUMP TO 7600 AFTER LOAD IS FINISHED
↑↑↑↑
```

```
    TYPE < CONTROL > P TO JUMP TO START ADDRESS.
    LOAD TAPE FOR PASS 2, TYPE < CONTROL > P.
    END OF PASS 1 TYPE < CONTROL > P
    LOADER IS WAITING FOR YOU TO LOAD A TAPE AND
    TYPE < CONTROL > P
```

c. Save the core image as described under SAVE/CALL in this document. Note the exclamation point (!) after the name. This means system program as opposed to the normal colon (:) of a user program save. (See Appendix A).

d. Repeat process from step b until you have stored all the desired systems programs.

# 4. OPERATION OF THE DISC SYSTEM

## 4.1 Loader

### 4.1.1 General

General – The loader is a disappearing program allowing paper tape or disc input. It loads in one or two passes and makes all of core available to the user, except the page reserved for the monitor head (7600-7777). The user specifies a command list to the loader and the entry point. After a successful load the loader disappears and enters at the specified starting address.

### 4.1.2 Operations

Operations – From the monitor, type LOAD and wait for *IN —. The loader uses either high or low speed paper tape as specified at build time by the answer to HIGH SPEED PAPER TAPE?. If you specified high speed paper tape either T: or R: is forced to the high speed reader; if you said no, both devices are forced to low speed. This applies only to the loader.

The loader will load up to five separate tapes. This is done by typing the appropriate command string in response to *IN —. For example,

"*IN — R:,R:,R:⟩ "

causes the loader to expect three separate tapes to be loaded. If the two pass option is specified, each of the three tapes must be read in twice. The loader acknowledges each valid input with a "<cr><LF>↑".

The next typeout is "*OPT —" which expects a 1-character answer 1 or 2. This argument is the number of passes required. The one pass loader will not load programs in field 0 above 7000. The loader disturbs core from 6000 to 7577. The two pass loader will load programs in all of field 0 except the cells described below.

| Core Size | Cells Excepted |
|-----------|----------------|
| 4K | 7600-7777 |
| 8-32K | 7574-7777 |

The two pass loader requires the input tapes be passed through the reader twice. In either case higher fields will be loaded. In the command string, each field, delimited by commas, contains device of one binary program and this command string will be identical for one pass or two pass load of the same program.

The loader asks "*ST = " and expects a one to five digit answer describing, in octal, the starting address. The highest digit of a five digit number describes the memory extension involved; a four digit or less string implies field 0. For example,

" ST = 30200⟩ "

means jump to field 3, address 200.

The loader then types "↑" and waits for a tape to be inserted in the reader. When the tape is positioned, type <CONTROL>P. The tape is read-in and the loader again types "↑" and waits. The low speed reader version of the loader halts; press Continue to resume the requested operation. Continue the above operation until you have matched the number of tapes loaded with the input request. The typing of one more <CONTROL>P causes the loader to enter at the specified starting address. Note that two <CONTROL>P's are required at the end of pass 1 of a two pass load. In two pass mode, tapes are loaded in the following order: Tape 1, Tape 2, Tape 3..., then Tape 1, Tape 2, Tape 3... .

4.1.3    Errors – An illegal checksum causes the loader to type "?" and return to the monitor.

4.2    CALL/SAVE

4.2.1    SAVE – Core may be saved on the disc in blocks of $128_{10}$ words. From 1 to 31 (decimal) blocks may be placed on the disc and the blocks may be contiguous or non-contiguous as specified by the user.

The syntax is as follows:

```
.SAVEₓXXXX{ : }YYYYY -ZZZZZ,WWWWW;UUUUU )
```

- ENTRY POINT
- ONE PAGE SAVE
- MULTIPLE PAGE SAVE
- : USER PROGRAM
- ! SYSTEM PROGRAM
- PROGRAM NAME
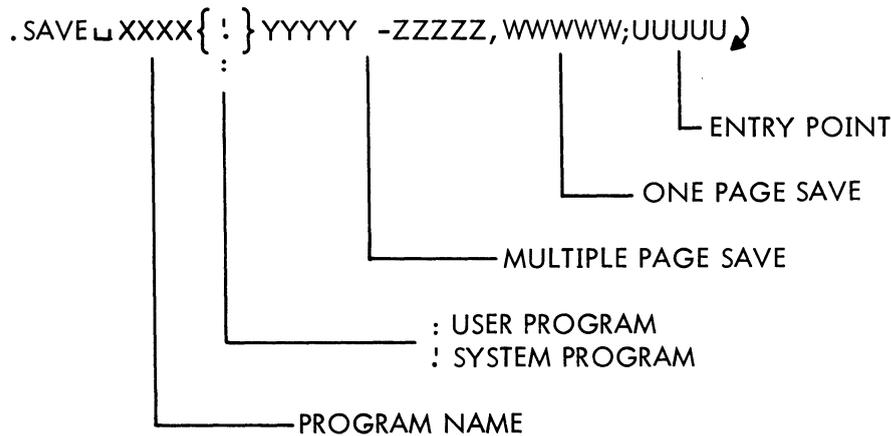
Any combination of multiple page SAVE and/or single page SAVE requests may be requested, up to 32 (decimal) requests. Beginning address of a multiple page request must be smaller than the ending address. All addresses must be in the same field. All addresses are in octal and are one through five digits where the most significant digit of five is the memory extension. For example,

.SAVE    CORE:    0 , 400-6177; 20 )

saves a core image called CORE, as page 0, and all cells from 400 through 6177 with an entry point of 20. Extended memory may be saved through the use of a five digit number. This most significant digit describes the field to be accessed and no check is made for validity in your system. If you save in field 7 you must have memory to match the request. The last non-zero extended field digit entered is remembered. The following three examples perform in exactly the same way.

.SAVE    FOO:10000 - 17777; 10200
.SAVE    FOO:30000 - 67777; 10200
.SAVE    FOO:10000 - 7777;   200

4

Only one field may be saved at a time, and an entry point of zero causes a return to monitor at call time.

4.2.2   <u>CALL</u> - Core may be loaded from the disc by name in the following ways:

.CALL  XXXX )

or    .XXXX

The former is used if a user program is desired.  The latter is used if a system program is desired (see " ! " and " : " in SAVE).  At the successful completion of the "call", control is transferred to the entry point specified at save time.  Only the core specified at save time is disturbed.

5.      DETAILS OF OPERATION AND STORAGE

5.1     Core Configuration

The system resides in the top page of field 0.  The entry point to the system I/O routine is 7642.  The loader, SAVE and CALL can access extended memory.  Two modes are used in the system; monitor mode, and user mode.  Monitor mode is present when ↑C is typed at system program time or when the monitor is started.  The " ." means monitor mode; the "*" means user mode.  (See Figure 1).
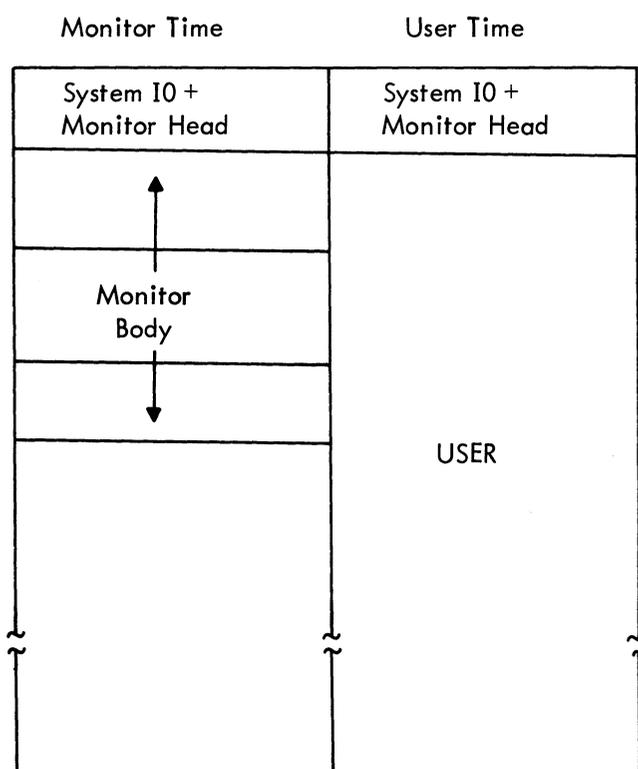


Figure 1   Core Storage Configuration

5

The operation of the system after it is built and stored on the disc is described below. Appendix B shows the bootstrap which must be toggled in to start the system, if the machine core is cleared or destroyed.

At monitor and SP time the keyboard listens to a ↑C and returns to the head of the monitor (top page of field 0). The system is completely open ended, allowing flexibility in system content at the user installation. The system is built easily on the user's machine from a tutorial system builder program.

Monitor start is at location 7600 of field 0. At monitor start the code from 7200 to 7577 is saved in the first two scratch blocks on the systems device, and the monitor (block 1 and 2) is read into these cells. Control is transferred to the monitor which types a carriage return, line feed and a ".". A "." at the start of any line informs the user that he is communicating with the monitor. A <CONTROL> C(↑C), when typed to either the monitor or loader causes an immediate transfer of control to 7600. A rubout typed to the monitor causes a monitor restart without destroying the contents of the scratch blocks on the systems device.

Core images may be saved by typing:  SAVE NAME: $n_1$ -$n_2$, $n_3$; $n_4$ ⟩ , where NAME is any one to four character program name. The dash between $n_1$ and $n_2$ indicates a contiguous save from the page containing address $n_1$ through the page containing $n_2$. Address $n_3$ is comma delimited and indicates a single page save. The entry point $n_4$ is delineated by a semicolon.

A list showing the required pages is constructed from the information typed by the operator and a block requirement count is kept. When the carriage return is typed (allowing the save to begin) a directory name (DN) search is initiated. If a file of the name specified is found, it is replaced by the present file; if not, a new file is created. A storage availability search finds sufficient available blocks to satisfy the requirement count generated as the page list was built. Their numbers are stored on a corresponding block list. These blocks are then filled with the contents of the pages to be saved until the typed request has been satisfied. When the save is completed control returns to the start of the monitor (7600).

A save file may be recovered by typing "CALL NAME ⟩". The file is loaded back into core and control is transferred to the entry point specified during the save. If the file was designated a systems program at save time it will be recovered by typing only NAME ⟩ . When a program is called, a DN search is initiated to find the name specified. Associated with the name is the entry point, and file protection and extension information. If the DN entry is found to be a save file of the proper mode (systems or non-systems), calling proceeds. If not, the call is terminated and control is returned to the monitor. When the call is completed (i.e., all of the saved code is back in core) control is transferred to the entry point specified at save time. The time for a program to be saved or called is less then the time to type the request.

## 5.2    Mass Storage Configuration

5.2.1    File Structure - A relatively sophisticated file structure is implemented to allow for automatic retrieval of storage for use by the system and a requirement for a single DECtape system.  (See Figure 2.) Two types of blocks are present - directory name (DN), and storage allocation map (SAM) blocks.  The former are described in Figure 3, the latter in Figure 4.  All blocks are linked with the end of chain line - 0.  The top of the SAM block chain is addressed in all DN blocks.  The SAM block is the item which allows automatic storage retrieval.  Each DN entry has a SAM number between $1-77_8$.  A zero means unused.  A SAM block has a record of $256_{10}$ pages of mass storage (Figure 4).  The first page of any numbered program is found by searching SAM blocks until an entry with matching number is found. To add to a file, find a SAM block entry equal to 0, insert the correct SAM number and adjust the storage block linking.  To delete a block, the DN entry is cleared and the SAM block entries (equal to the DN number) are zeroed out.

| |
|---|
| DIRECTORY (BACK-UP) |
| MONITOR |
| LOADER |
| FORTRAN |
| EDITOR |
| DDT |
| OTS |
| PIP |
| ASSEMBLER |
| DIRECTORY (NORMAL) |
| USER |
| SCRATCH      256 |

16K
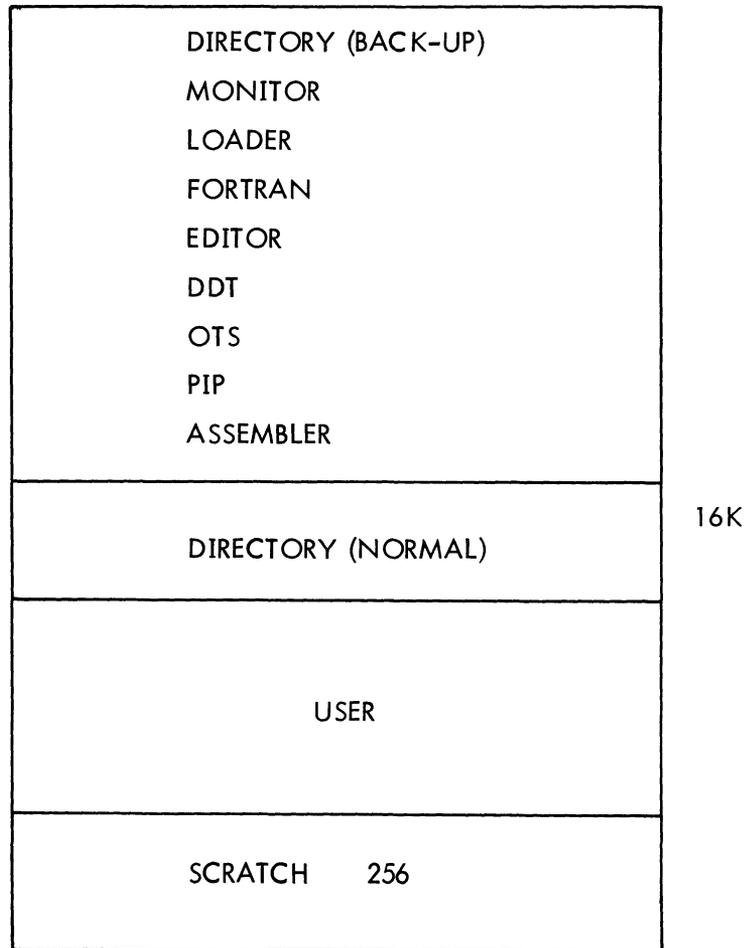
Figure 2   Mass Storage Configuration

| 0 |
|---|
| 0 |
| FIRST SAM BLOCK = # = NUMBER |

| N | A |
|---|---|
| M | E |

◄— 4 CHARACTER NAME (SIXBIT)

NAME BLOCK ENTRY

START        ADDRESS

ENTRY        POINT

Extended Memory Bits —

PROG. TYPE
00 Source
01 Binary
10 Fort. Binary
11 Save

25 ENTRYS/BLOCK

1 = System Program

LINK
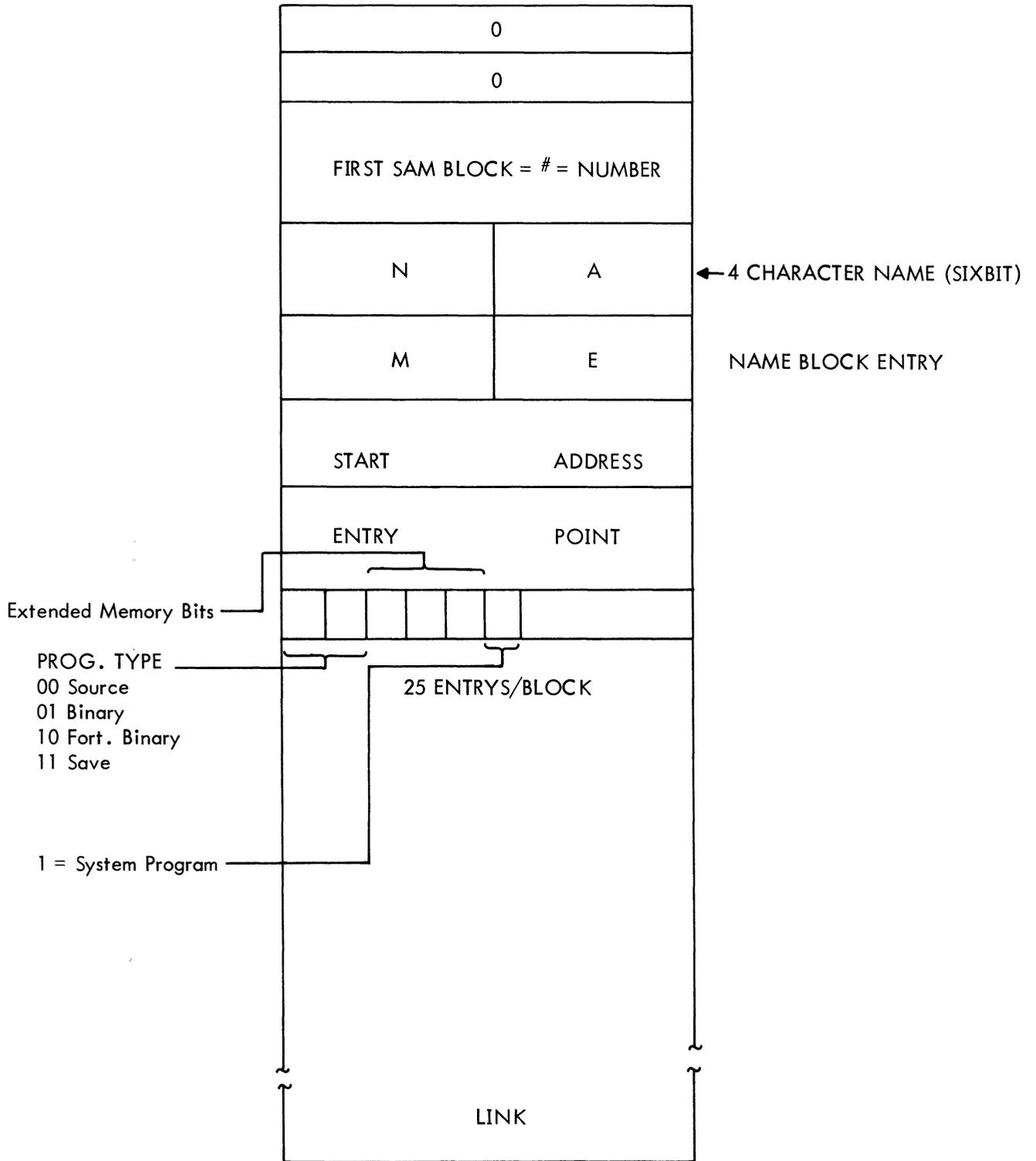
Figure 3   Directory Name Block

SAM BLOCK LAYOUT

SAM BLOCK IMAGE

| Block 200 | Block 0 |
|-----------|---------|
| Block 201 | Block 1 |
| Block 202 | Block 2 |
| Block 203 | Block 3 |
|           | Block 4 |

| | |
|---|---|
| 0 | 1 |
| 0 | 1 |
| 0 | 1 |
| 1 | 0 |
| 1 | 4 |
| 0 | 0 |
| 0 | 3 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 10 | 4 |
| 0 | 3 |
| 0 | 3 |
| 0 | 4 |

Block 377 | Block 177

LINK

LINK

Link to SAM Block 2 w/
Block 400 - 577
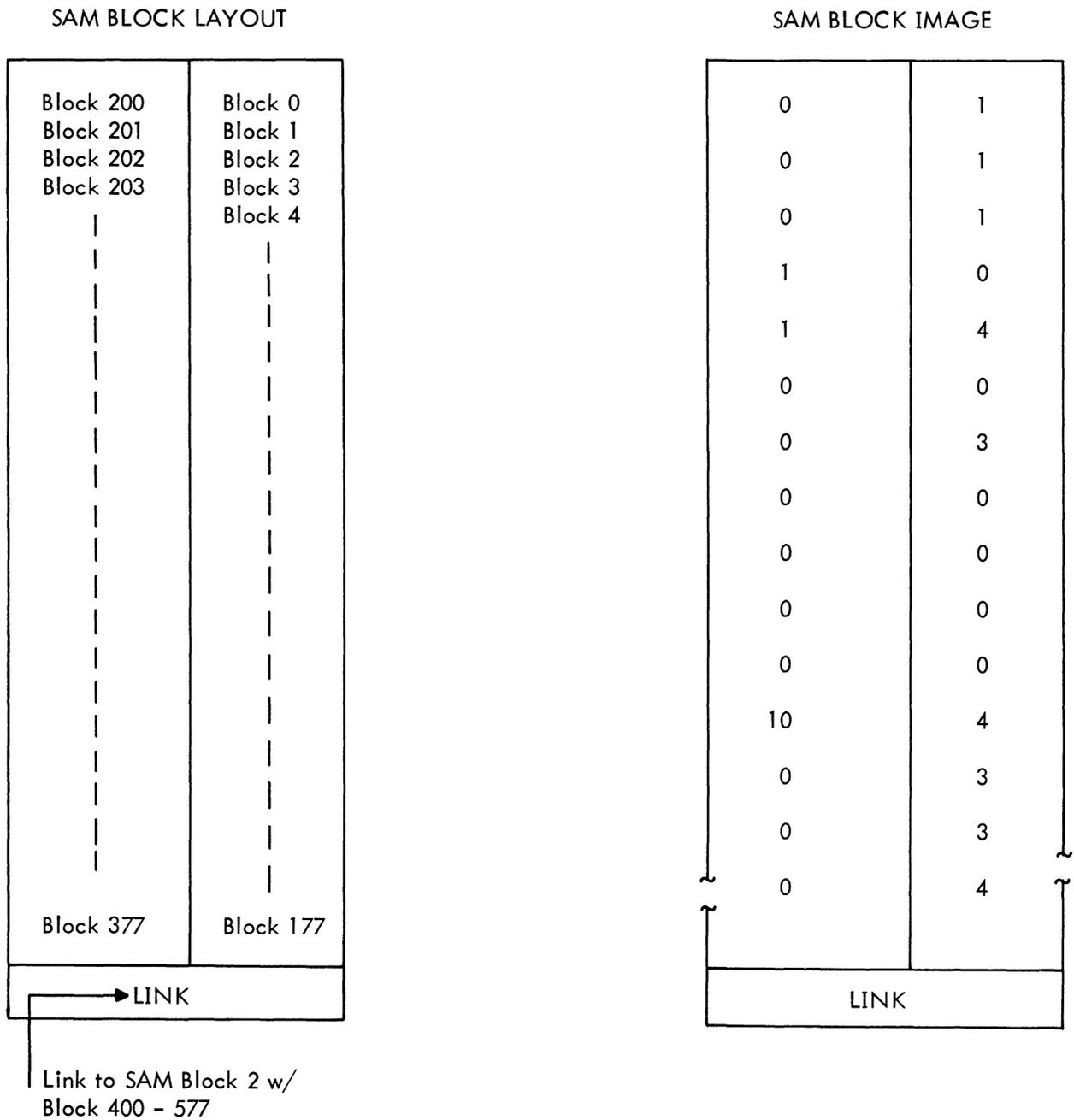
Figure 4    Storage Allocation Map

5.2.2    Disc Configuration - The Disc configuration is shown in Figure 5B. The scratch area for system overlay is located in the last two pages of disc storage. Additional DN and SAM blocks are added, one each per 32K of disc.

5.2.3    DECtape Configuration - The DECtape configuration (Figure 5A) is logically equivalent to the disc system but has the SAM and DN blocks ordered differently for efficiency.

BLOCK

| DEC TAPE (A) | BLOCK | DISC (B) |
|---|---|---|
| HEAD OF MONITOR | 0 | HEAD OF MONITOR |
| MONITOR | 1 | MONITOR |
| MONITOR | 2 | MONITOR |
| DN | 3 | DN |
| DN | 4 | SAM |
| SCRATCH | 5 | SYSTEMS |
| SCRATCH | 6 | |
| SCRATCH | 7 | |
| SAM | 10 | |
| SAM | 11 | |
| SAM | 12 | |
| SAM | 13 | 16K |
| SYSTEMS | | |
| | | DN |
| | | SAM   1 per 32K additional disc |
| | | USER |
| | | $DN_2$ |
| USER | | $SAM_2$   $SAM_3$ $SAM_4$ |
| | | USER |
| | | SCRATCH |
| | LAST | SCRATCH |
| | | SCRATCH |
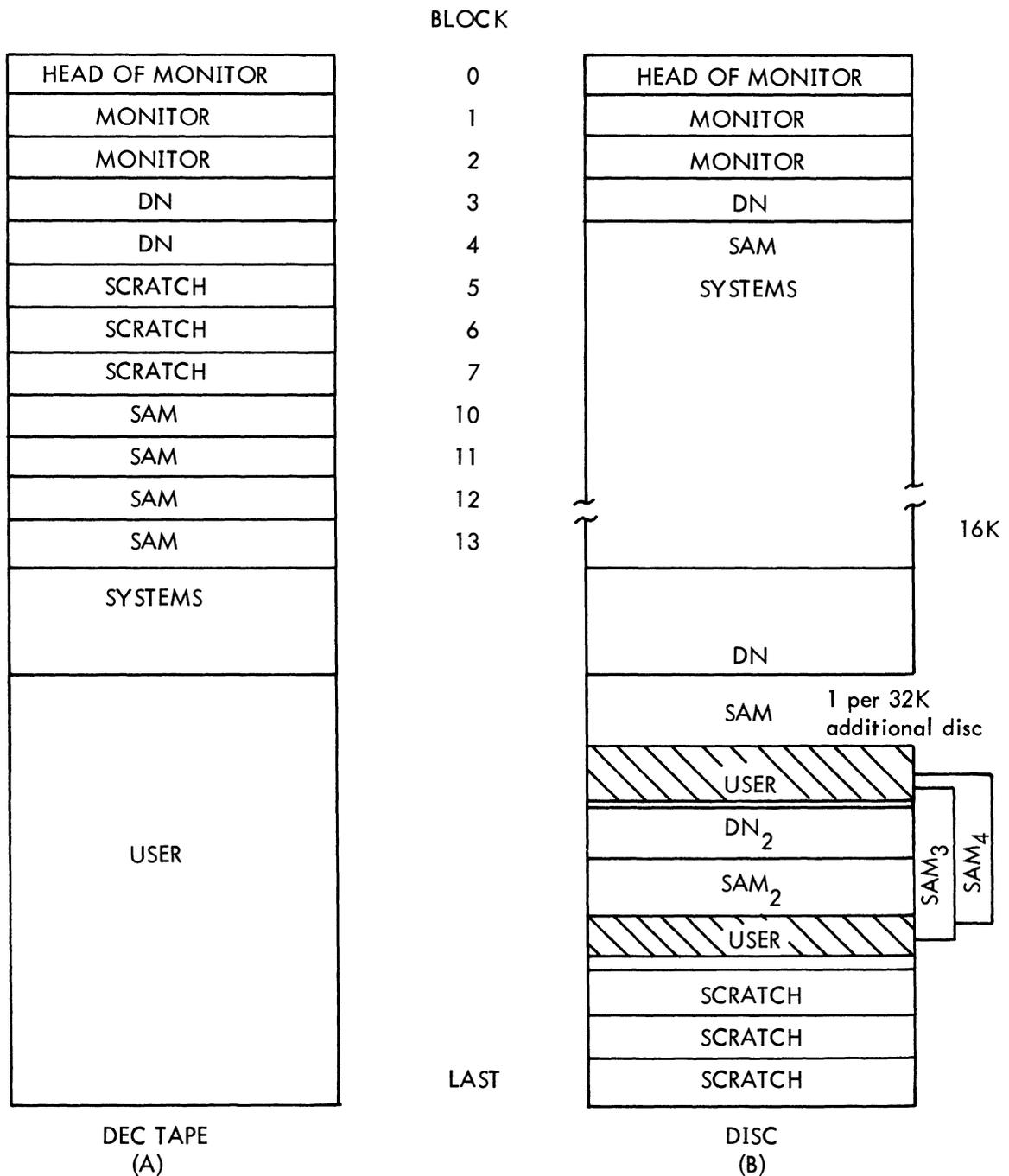
Figure 5  Mass Storage Configuration

5.3      Error Messages

a.    The monitor protects the user from himself to the extent that it recognizes incorrect syntax in input command strings.  Any errors discovered are remembered and at the time the user types a carriage return the monitor signals them with a "?".

b.    The command decoder notifies the user of the following errors, by typing the key letter shown opposite the offending condition.

| | | |
|---|---|---|
| No such inputs | – | I |
| Illegal input characters or<br>  miscellaneous error<br>  condition | – | ? |
| Directory full | – | D |
| System I/O failure | – | S |
| Excessive inputs or outputs | – | E |

c.    The local errors are described in the write-ups for each system program.

d.    Monitor time read/write errors cause a halt to occur.  Persistence of this condition means a hardware failure, as the system I/O routine tries to read and write three times before it gives up.

6        SPECIAL REQUIREMENTS INSTRUCTIONS OR FORMATS

6.1      Command Strings

The command string structure for the PDP-8 Disc System software enables the user to easily communicate with the system.  The punctuation of the input is as described below.

| | |
|---|---|
| . | = monitor mode |
| * | = system program mode |
| , | = more devices in list |
| : | = separator between device and file name |
| ↑C | = <CONTROL> C Request to return to monitor |
| ↲ | = carriage return |
| ↑P | = <CONTROL> P.  Proceed with next operation. |
| Rubout | = <RO> Go back to start of input string. |

6.2      Device Names

| | |
|---|---|
| DX | – DECtape if disc and DECtape are on system [ X = unit number 0-7] * |
| S | – System (disc or DECtape) |
| R | – High-speed paper tape |
| T | – Teletype (low-speed paper tape) |

* Not implemented

The command decoder used by all system programs asks one to three questions:

    a.   OUTPUT in the form "*OUT-".

    b.   INPUT in the form "*IN-".

    c.   OPTIONS or SWITCHES in the form "*OPT-".

Each system program may chose to use any or all of these options when it enters the command decoder. The user may type in one output device, one option (any character, which is transmitted in ASCII to the using program) and a string of one or more input devices.

Each system program using the command decoder must tell it the type of file being dealt with, i.e., Binary, Fort. Bin., or ASCII.

Some examples of command strings are shown below.

       *IN — S:SAM ⟩

       *IN — S:FOO2,R: ⟩

       *OUT — D7:BAR ⟩

       *OPT — M ⟩

```
*TYPE SIZE OF CORE (IN K)
*4
*HIGH SPEED PAPER TAPE?
*YES
*DISC OR TAPE?
*DISC
*TYPE NUMBER OF DISC UNITS
*1
*
.LOAD
*IN-R:
↑
*OPT-2
ST = 7600
↑↑↑↑
.SAVE FORT! 0-7577;4511
.LOAD
*IN-R:
↑
*OPT-1
ST = 7600
↑↑
.SAVE PAL! 0-3400;241
.LOAD
*IN-R:
↑
*OPT-2
ST = 7600
↑↑↑↑
.SAVE DDT! 5000-7577;5400
.
```

Typical System Builder Output

Figure 6

12

## APPENDIX A
## SAVE SYNTAX FOR STANDARD PDP-8 SYSTEM PROGRAMS

```
.SAVE  FORT!  0 - 7577; 4511
.SAVE  PAL!   0 - 3400; 241
.SAVE  DDT!   5000 - 7577; 5400
.SAVE  LODT!  1000 - 1577; 1000
.SAVE  HODT!  7000 - 7577; 7000
.SAVE  MAC8!  0 - 7577; 2041*
.SAVE  FOTS!  0 - 6200; 200
```

* LOAD 200, SET SWITCHES + RUN.

## APPENDIX B
### BOOTSTRAP LOADERS

A.  DISC BOOTSTRAP

| | | |
|------|--------|------------|
| 0200 | /7600  | /CLA |
| 0201 | /6615  | /DEAL |
| 0202 | /1215  | /TAD 215 |
| 0203 | /3613  | /DCA I 213 |
| 0204 | /1212  | /TAD 212 |
| 0205 | /3614  | /DCA I 214 |
| 0206 | /6603  | /DMAR |
| 0207 | /6622  | /DFSC |
| 0210 | /5207  | /JMP 207 |
| 0211 | /5600  | /JMP I 200 |
| 0212 | /7576  | |
| 0213 | /7750  | /WC |
| 0214 | /7751  | /CA |
| 0215 | /7577  | |

B.  DECTAPE BOOTSTRAP

(See TC01 System Bootstrap)

## APPENDIX C
### CALLING SEQUENCE (SYSTEM ROUTINE)

A     CALLING SEQUENCE

| | |
|---|---|
| JMS I SYSIO | / CELL "SYSIO" POINTS TO I/O |
| FUNCT | / SEE PARA. II |
| BLOCK | / BLOCK TO BE ACCESSED |
| CORE | / LOW ORDER CORE ADDRESS |
| LINK | / FILLED BY WRITE, USED BY READ |
| ERROR | / ERROR RETURNS HERE |
| . | / NORMAL RETURN HERE |
| . | |

B     FUNCTION CELL

| | |
|---|---|
| BITS 0-1 | UNUSED |
| BITS 2 | =0, NORMAL; =1, RETURN AT END OF READ/WRITE TO (ADDRESS + 1 IN ERROR RETURN. |
| BITS 3-5 | UNIT NO. (IF DECTAPE) |
| BITS 6-8 | MEMORY FIELD |
| BITS 9-11 | FUNCTION; READ = 3, WRITE = 5. |