DEC STD 125 REV. B

CASSETTE FORMAT FOR FILES

TITLE: CASSETTE FORMAT STANDARD FOR LABELLED AND UNLABELLED FILES

ABSTRACT: Describes the format and labelling conventions for files, physical blocks, logical records and data written on Digital Equipment Corporation Cassettes. It also describes the unlabelled standard. This standard must be followed when reading and writing cassettes intended for interchange between systems; it is recommended for other cassettes.

This standard has been reaffirmed without change by the Software Standards Approval Committee on 18 June 1981.

FOR INTERNAL USE ONLY

DATE	ECO #	ORIGINATOR	APPROVED	REV
21-Feb-75	--	D. Pavlock	R. Amann (Eng. Comm.) 06/05/75	A
18-Jun-81	ML001	P. White	Eng. Comm.	B

Document Identifier

Size	Code	Number	Rev
A	DS	EL00125-00-0	B

NP1-005

digital

TABLE OF CONTENTS/REVISION STATUS

TABLE OF CONTENTS/REVISION STATUS (Cont'd)

1.0 INTRODUCTION

1.1 MOTIVATION

This standard was written to provide a measure of compatibility among Digital's systems that support cassettes. It was done at this time because Digital now supports this medium.

1.2 GOALS

This standard is intended to (1) furnish a design for labelled files that will allow the users to write files on one system that supports cassettes, and read them on another, (2) provide a consistent growth pattern for support of cassettes, through a system of levels of support, (3) allow the fewest number of data formats consistent with the needs of computers with different word lengths, and (4) provide a standard for unlabelled files.

1.3 SCOPE

This standard applies to all Digital Software products that support labelled cassette files. The following products currently support Digital cassette:

CAPS-8
CAPS-11/CAPS-11 BASIC
DOS-11
OS/8
RT-11
RSX-11M

This standard implies changes for each of those systems.

Other systems that wish to support cassettes must follow this standard.

Cassette hardware is not required to put out a file gap at the beginning of the tape. Future cassette drivers and controllers must not prohibit software from writing at least level zero cassettes.

1.4 HISTORY

1.4.1 Deletions

One previous effort was made at standardizing file formats. It allowed for several items that have been deleted from the current standard; variable-length records, an optional second header record, and several extra data formats. Variable length records were eliminated, because they are hard to implement. The supplementary header records were rejected, because the size of the medium seemed to make a second header a candidate for overkill. We eliminated some data formats because we decided on a goal of a minimum number of data formats.

1.4.2 Change To Previous Standard Proposal

The codes for the binary data formats were redefined, because the previous standard did not adequately spell out how the codes were to be used, and confusion resulted. Thus, code zero and codes in the range 2 to 14 (used by the previous standard proposal) are now listed as undefined. In order to avoid the necessity for customer conversion, a system encountering cassettes with a code in that range should assume correct type and continue. Example: A DOS-11 Linker would ordinarily expect data type 22. If it encounters data type 3, it assumes correct format, and continues.

1.4.3 Additions

The previous effort was not rigorous in its definition of levels. Ther was a minimal subset, and then several options beyond that. In that the primary goal was to ensure the possibility of using cassettes as an interchange medium, we decided to strictly regulate the way in which support for cassettes could be increased. Also added were a generation number and definitions of formerly unused bytes in the header as reserved for (1) future standard use, and (2) user use. Cassette labels (as opposed to file labels) were considered and rejected for size considerations.

All levels are nested, i.e., if the operating system claims level n cassette support, this implies it supports all features of level n and levels zero through n-1. Therefore, it is necessary that all level-zero reading programs be able to read level zero format and that all level-zero writing programs shall write only level-zero format. Further, that all level-one reading programs be able to read level-zero and level-one formats and that all level-one programs shall write only level-one format including at least one level-zero format. Further, that all level-two reading programs be able to read level-two format including at least one level-zero format. All level-zero formats must be strict subsets of level-one formats. All level-one formats must be strict subsets of level-two formats. This implies that any system which supports DEC cassettes must have the ability to read, write and zero cassettes containing type 1 ASCII files. Therefore, level zero, is the only guaranteed interchange level.

1.5 RELATED STANDARDS ACTIVITIES

ANSI and ISO have recently formed committees to work in this area and may generate standards which may have to be dealt with in future revisions of this standard.

1.6 FUTURE STANDARDS ACTIVITIES

It is expected that ASCII and binary data must comform to the DEC standard ASCII and Binary formats, when and if such formats are defined.

Designers wishing to define new data formats should remember the more formats, the harder becomes compatibility. If it still appears that a new format is required, the proper procedure is to petition the Software Engineering Standards Committee.

2.0 TERMINOLOGY

A CASSETTE consists of a sequence of one or more FILES, separated from each other by a single FILE GAP. The first file on the cassette must be preceded by a file gap*; the last file must be followed by a file gap and a SENTINEL FILE (see Section 3.2), or by clear trailer.

Each file consists of a sequence of a file header block plus zero or more data blocks, separated from each other by block gaps. The first block of a file is called the file header block, or file label.

A block consists of a sequence of 1 to 2**16 - 1 DATA BYTES followed by a 2-byte CYCLIC REDUNDANCY CHECK. (This is a logical limit, there is no physical limit, except for the length of the tape.)

A cassette BYTE is eight bits (binary number). A BIT is a binary zero (0) or one (1).

A CHARACTER is a byte interpreted via the ASCII character codes. Parity is not required. The high order bit (bit 7, the leftmost bit) of each eight bit byte containing an ASCII character should be masked on reading. Parity is checked by the software only, not by the hardware.

A GENERATION NUMBER is a number assigned to a file at creation, to distinguish one file from a previous version of the same file.

Blank — The ASCII character 'space', whose value is 040 (octal).

Null — The ASCII character whose value is 000 (octal).

Zero Byte — A byte all of whose bits are zeroes.

A File Key is defined as n number of characters of file name and m number of characters of file name extension (see Section 3.0 for definition of File Key for each level).

File names and extensions must consist of letters, numerals and blanks 40 (octal). The first character must not be blank; there shall be no imbedded blanks within the name or extension; and short names must be padded on the right with blanks. For level two, bytes 0-5 and 26-28 are considered as a unit, when applying these rules. For levels zero and one, bytes 26-28 are undefined.

Cassette ASCII is defined as seven bit ASCII, bit 8 is undefined and is ignored on reading.

---

* The software must ensure the existence of this initial file gap, by requiring hardware that does it automatically (e.g., the TA11) or by writing its own.

End of file is defined as:

a. Spare bytes in the last block are filled with nulls followed by a file gap

or

b. CTRL/Z (Ø32(8)) and all data following must be ignored.

3.Ø THE STANDARD - LEVELS

There are three levels for the labelled standard:

LEVEL ZERO:

Level Zero must support:

1. 32 (decimal)-byte header block, which contains:

   a. six-character name of file
   b. three-character file name extension
   c. date
   d. data-type indication
   e. eight-bit binary generation number

2. Logical end-of-file

3. Logical end-of-tape

4. Fixed-length, 128 (decimal)-byte blocks

5. ASCII data (type 1)

6. Optionally, any other listed data type

7. File Key is defined as a six character file name and first two characters of extension.

LEVEL ONE:

Level One must support:

1. All attributes of level zero

2. Read/write support for multi-volume files

3. File Key is defined as a six-character file name and three characters of extension.

LEVEL TWO:

1. All attributes of level zero
2. All attributes of level one
3. 9-character file names
4. 16-bit binary generation number
5. A record attributes byte
6. Fixed-length blocks of from 1 to 2**16-1 bytes in length
7. File Key is defined as a nine character file name and three characters of extension.

3.1 THE FILE HEADER BLOCK

Each labelled file must begin with a 32 (decimal)-byte file header block. Section 5.1 illustrates the format of the file header block.

3.1.1 The File Name

The name is in 7-bit ASCII. The eigth bit is undefined, and must be masked off on reading.

Multiple files with the same file key (for a given level) shall not appear on a single cassette intended for interchange. If the software system does not enforce this, the user manual for the software/hardware system must instruct the user to generate unique file keys, if this cassette is intended for interchange.

NOTE

> A file may be logically deleted by changing the name to *EMPTY. To check for a deleted file, check only the first character. This is to allow for future means for deleting a file (e.g., *BAD).

3.1.2 The Data Type

Ability to read and write ASCII data (type 1) is required for level zero. Any system may support any other data type, regardless of level, however these data types are not required to be supported for interchange purposes.

Byte 9 in the file header block contains the "Data Type". The Data Type defines the way in which data is recorded in that file. Table 3-1 lists the Type Codes and gives the meaning associated with each.

The goal is to have the minimum number of types consistent with systems having different word length. Odd numbered types are reserved for ASCII types, to allow a single bit to show the presence of ASCII data.

Type zero (undefined) is required, because files on OS-8 and RT-11 disks do not carry data type information, but file transfers between disks and cassettes should not be prohibited for that reason. Hence, for example, the OS-8 MCPIP program will transfer disk files to cassette and give a zero type, unless the user specifies type. Paragraph 1.3.2 describes the reasons for omitting definitions for types 2 through 14.

For further explanation of the various definitions of these data types, see Appendix A.

Table 3-1 Standard Data Types

Type	Description
0	Unknown data type. To copy a file of this type to another medium, copy all 8 bits per byte and store in a format that can be restored to a cassette.
1	Cassette ASCII.
2-14	Unknown data type. To copy a file of this type to another medium, copy all 8 bits per byte and store in a format that can be restored to a cassette. (These codes must not be used by any new software.) See Appendix C for explanation of the use of these codes in old software systems.
15	ASCII characters with line numbers. To be specified.
16	PDP-8 Cassette bin-loader format. Defined in the CAPS-8 User Manual. No other PDP-8 format may use this type code.
17	Reserved for DEC standard ASCII, when and if it is specified.
20	PDP-11 OBJ format. Not to be used for other PDP-11 binary formats.
22	PDP-11 LDA format. See comments on Type 20.
24	Reserved for future use of PDP-10.
26	PDP-11 TSK format defined in RSX-11M Task Builder Manual.
30	Bootstrap File for PDP-8.
32	Bootstrap File for PDP-11.
34-50	Reserved for bootstraps.

If the type is known, all programs must set the type byte correctly. They must write zero (0) if type is unknown. Programs reading files may check this type to see if it agrees with the expected type and give a warning message on disagreement.

3.1.3 File Block Length

Bytes 10 and 11 of the File Header Block contain the length of each (non-header) block up to the next file gap. Level zero and one requires 128 here; level two files must have any non-zero value.

NOTE

> Byte 10 contains the most significant bits. Thus, the record length equals (256) 10 times the contents of byte 10 plus the contents of byte 11.

If records in this file have variable lengths, byte 10 and 11 must contain zeroes. Such a file violates the standard.

3.1.4 File Sequence Number

This byte is for multi-volume files. Byte 12 is undefined for level zero files. It must be a zero byte for single-volume level one and two files, or the first volume of multi-volume file. Successive continuation files on different cassettes should be numbered 1,2,3,... in this field.

3.1.5 Level

Level zero files must insert a zero byte in this position; level one files a one, and level two files a two. The high-order four bits in this byte are reserved for the possibility of continuation header blocks. Thus programs must ignore the high-order four bits when checking level number. This is reserved for future standards use.

3.1.6 File Creation Date

This field is required for all levels. The file creation date is contained in the six characters starting at byte 14. When specified, this date shall consist of six 7-bit ASCII characters specifying the day number (01-31), the month number (01-12) and the last two digits of the year number in the order ddmmyy. If no date is specified, the first byte will be zero (null) or blank (40)8. Date must be inserted, if known.

3.1.7 Generation Number

Byte 2Ø contains an 8-bit binary generation number for all levels. It must be zero if the generation number is unknown or not supported (as in CAPS-11 or DOS/BATCH-11). Level two files have a 16-bit generation number, with high-order bits in byte 2Ø. Byte 21 must be zero for level zero and one files.

3.1.8 Record Attributes

Byte 22 specifies certain characteristics for data recorded on level two files. Bit Ø refers to formatting of records destined for printing device, i.e., line printer, terminal, etc. The definition of this byte is:

Bit Ø - If 1, indicates that when printing the data, the first character of the record is to be interpreted as FORTRAN carriage control character.

Bits 1-7 - Unused; must be zero.

This byte is undefined for levels zero and one.

3.1.9 Unused Bytes

Bytes 23-25 are undefined for levels zero and one; they are reserved for future use by the standards, and must be set to a null for level two.

3.1.1Ø Extended Filename Bytes

Level two files insert the last three characters of the filename in byte 26-28. Contents of these bytes must be 7-bit ASCII characters for level two. These bytes are undefined for level zero and level one.

3.1.11 User Bytes

Bytes 29-31 are reserved for the user. Default must be written as zero or as supplied.

3.2 SENTINEL FILE

Logical End of Tape may be denoted by clear trailer or a 32-byte file header block with the first byte null. Such a block follows a file gap and is called a sentinel file. See Section 5.3 for examples of logical End of Tape.

3.3 BOOTSTRAP FILES

Bootstrap files must be the first file on the tape, and must have exactly level zero characteristics, i.e., 128-byte blocks, file names, etc. Data type must show bootstrap type.

3.4 MULTI-VOLUME FILE SUPPORT

Level zero systems do not support multi-volume files.

3.4.1 READ Support

Level one and two systems should always check byte 12 of the header block for the expected value. When a file is opened, the expected value of the first volume is zero. The number for each successive volume is incremented by one. If the expected value is not found, the system must give a warning.

Level zero systems and other systems reading level zero cassettes must report end of file when clear trailer or a file gap is reached during READ. Level one or two systems reading level one or two cassettes must report end of file only on reaching a file gap. When they reach clear trailer, they must output a message to the operator asking whether end of file has been reached.

The operator must indicate whether more volumes exist. If the reply indicates no more volumes, the system must react as though end of file were reached. If the reply indicates another volume, the system must allow the operator to load the cassette. It must then search the cassette for a file with the same name and version number as the last one, and the next higher volume number (byte 12). If it finds such a file, the system must then continue processing. If no such file is on the cassette, the system must report same and allow the operator to mount another cassette.

3.4.2 WRITE Support

Level zero systems must give a "device full" message when clear trailer is reached, and close the file. (This implies that the last file on the cassette may be an incomplete one.) Level one and two systems that reach clear trailer on WRITE must:

1. Insure that the block being written when clear trailer was reached will give a clear trailer error when read back. This involves first checking the byte transfer count to determine if all bytes of the current block were transferred to cassette before the clear trailer indication was received. If the count indicates that all bytes were not transferred, then this partially-written block will always give a clear trailer indication when read back with the proper block size. This last block must be retained for transfer to the next volume, if the operator so specifies. If the byte transfer count indicates that all bytes were successfully transferred to cassette before the clear trailer indication was received, then the system must backspace one block and write a file gap over this last block. Writing a file gap if all data bytes were transferred insures that this block cannot be read on any drive, and thus, the block will not be duplicated if the operator chooses to continue the file on another volume. This last block must be retained for transfer to the next volume if the operator so specifies.

2. Send a message to the operator indicating that physical end of tape has been reached and requesting that the operator mount another volume. The operator must have the option of closing the file without mounting a new cassette. (This always results in the loss of at least the last block of the file.) If the operator wishes to close the file, the system should rewind the volume which filled up. It need perform no further I/O on this volume; the last file is already effectively closed. If the operator wishes to continue the file on another volume, he should remove the volume which filled up and mount another volume on the same drive.

3. After the operator has loaded the cassette, the system must space to logical end of tape, and write a new header with incremented volume number. It must then write out the block left over from the previous volume and continue processing. It is recommended that the operator have the further option of specifying that the newly-mounted volume be treated as a blank cassette. In this case, the system begins writing the header of the new file at the beginning of the tape (after the initial file gap). It does not space to logical end of tape.

NOTE

The multi-volume file write support for level one and two systems described above is intended only for use with fixed-length 128 byte blocks. For longer block lengths, writing a file gap over the block which was being written when clear trailer was detected can result in a gap large enough to be recognized by hardware as a file gap. Upon reading this last file on the volume, such a gap would signify logical end of file, even though the user may have specified that output be continued onto another volume. This problem does not arise with block lengths of 128 bytes or less, since overwriting a 128-byte block at physical end-of-tape with a file gap cannot result in a gap in a gap large enough to be recognized by hardware as a true file gap.

4.0 STANDARD FOR UNLABELLED CASSETTES FOR INTERCHANGE

Simple systems (e.g., "intelligent" terminals) may be able to support cassettes only in a manner similar to paper tape support. In such cases, they merely write data to the cassette in such a manner that it may be read back later. The cassette, in such cases, contains no files, no file block headers, no sentinel file, etc.

The format for an unlabelled cassette is as follows: first, the clear leader; next a file gap; then the data, grouped with successive fixed-length blocks. These blocks are separated by block gaps and the data is terminated by a file gap. Block length and content must be agreed upon in advance by systems wishing to interchange unlabelled cassettes. (128)10-byte blocks are recommended and shall be the default size.

Unlabelled cassettes are not recommended for interchange, since data formats are unspecified.

5.0 EXAMPLES

5.1 HEADER BLOCK FORMAT

The following diagram illustrates the format of the standard file header block. Detailed descriptions of the fields are contained in Section 3.1 of the standard.

Significance	Byte	Field	Type
	0–5	File name	
	6–8	File name Extension	A
	9	Data Type	B
most significant -->	10	Block Length	B
least significant -->	11		
	12	Sequence Number	B
	13	Support Level	B
	14–19	Date	A
most significant -->	20	Generation Number	B
least significant -->	21		
	22	REC Attributes	B
	23–25	Undefined for Levels 0 & 1, must be set to zero by level 2	B
	26–28	Last 3 characters of File name Level 2	A
	29–31	Reserved for Customer	B

Type:
A = ASCII
B = Binary

5.2 TYPICAL FILE HEADER BLOCK FOR LEVEL ZERO

```
---------------------------------------------------------------------
|        |     |   |   |      |   |   |         |          |             |
| FILNAM | TXT | 1 | Ø | 128  | Ø | Ø | Ø10173  | Ø      Ø | Unspecified |
|        |     |   |   |  1Ø  |   |   |         |          |             |
|        |     |   |   |      |   |   |         |          |             |
---------------------------------------------------------------------
Ø <-----> 6 <---> 9 1Ø   11    12  13  14        2Ø<---->25 26<------->31
```

FILENAME:	FILNAM.TXT
Type:	ASCII (1)
Block Length	128(1Ø)
Sequence	Ø = Multi-volume files not in level zero.
Level:	Zero
Date:	January 1, 1973
Generation	Number Zero

5.3 LOGICAL END OF TAPE

```
          ------------------------------------------------------
                   |              |              |              |
                   |              |              |              |
                   |              |              |              |
          ------------------------------------------------------
 Date Block         File Gap       Sentinel File

   Data     Block    Date    Block    Data       Block     Clear
   Block    Gap      Block   Gap      Block      Gap       Trailer
          ------------------------------------------------------
          |    |          |    |             |       |          |
          |    |          |    |             |       |          |
          |    |          |    |             |       |          |
          ------------------------------------------------------

          ------------------------------------------------------
          |    |               |             |       |          |
          |    |               |             |       |          |
          |    |               |             |       |          |
          ------------------------------------------------------
   Data     Block     Data            Block    Partially   Clear
   Block    Gap       Block           Gap      Written     Trailer
                                               Data Block
```

5.4 BEGINNING OF TAPE

```
 Clear    File                   Block   Data  Block   Data   File
Leader    Gap                     Gap   Block   Gap   Block    Gap
-----------------------------------------------------------------------
         |         |             |    |         |    |          |    |
         |         |             |    |         |    |          |    |
         |         |   HEADER    |    |         |    |          |    |
         |         |             |    |         |    |          |    |
-----------------------------------------------------------------------
                   <------------->    <--------->
                      32 Bytes          n bytes
                                     (128 for Level 0)
```

APPENDIX A

A.1 FORMAT OF PDP-8 BOOTSTRAP FILE

File must be such that when positioned as the first file on a cassette, it can be read in by the MI8-EL ROM bootstrap loader and when branched to will perform some bootstrap function (like read in the second file on the cassette which may be the monitor).

In order to create a bootstrap file properly, one must know what the 32-word ROM bootstrap does. A source listing is given below:

```
                                     /CASSETTE SYSTEM BOOTSTRAP

         /       COPYRIGHT 1972
         /       DIGITAL EQUIPMENT CORPORATION
         /       MAYNARD, MASS.  01754

         /       S.R.

         /STARTING LOCATION (NORMALLY):  4000

         6700             KCLR=6700
         6701             KSDR=6701
         6702             KSEN=6702
         6703             KSBF=6703
         6704             KLSA=6704
         6705             KSAF=6705
         6706             KGOA=6706
         6707             KRSB=6707
         7002             BSW=7002     /PDP-8/E, -8/F, AND -8/M ONLY
         3602             LOC=3602     /LOCATION WHERE SECONDARY
                                       /BOOTSTRAP REALLY GETS LOADED

         4000             *4000        /INITIALIZE PULSE CLEARS THE LINK
04000    1237    START,   TAD M50      /CHANGE READ CRC CODE (6) TO
                                       /REWIND <1>       [BIN]
04001    1206    CRCCHK,  TAD L260     /LOAD READ CRC CODE INTO STATUS
                                       /REGISTER A       [JMP I START]
04002    6704             KLSA         /FIRST TIME THROUGH, LINK MUST
                                       /BE 1 HERE
04003    6706             KGOA         /INITIATE THE OPERATION (READ
                                       /CRC OR REWIND OR FRWD FILE GAP)
04004    6703             KSBF         /READY?
04005    5204    RDCOD,   JMP .-1      /NO, WAIT
04006    7264    L260,    CML STA RAL  /SET L=1 AND AC= A HALT (7776)
04007    6702             KSEN         /ANY ERRORS?
04010    7610             SKP CLA      /NO
04011    3211             DCA          /HALT ON ANY ERROR EXCEPT FOR
```

```
                                        /REWIND OR FRWD FILE GAP
04012   3636          DCA I PTR         /CAN'T ALLOW 'TAD I PTR' LATER
                                        /TO AFFECT LINK
04013   1205          TAD RDCOD         /GET CODE FOR READ (0)
04014   6704          KLSA              /LOAD INTO STATUS REGISTER A
04015   6706   LOOP,  KGOA              /FIRST TIME STORES 173 INTO MEMORY
                                        /(8-BIT COMPLIMENT OF RDCOD)
                                        /OTHER TIMES READS ONE 6-BIT
                                        /BYTE OF PAIR
04016   6701          KSDR              /NEW DATA WORD READY?

04017   5216          JMP .-1           /NO, WAIT
04020   7002          RSW               /MOVE 6-BIT BYTE TO H.O. AC
04021   7430          SZL               /WHICH 6-BIT BYTE OF THE PAIR?
04022   1636          TAD I PTR         /2ND.  SO ADD IN 1ST BYTE
04023   7022          CML HSW           /SWAP BACK AGAIN.  SET LINK TO
                                        /INDICATE NEXT BYTE
04024   3636          DCA I PTR         /STORE BACK INTO MEMORY
04025   7420          SNL               /ARE WE DONE LOADING BOTH 6-BIT
                                        /BYTES?
04026   2236          ISZ PTR           /YES, SO POINT TO NEXT MEMORY WORD
04027   2235          ISZ KNT           /BUMP COUNTER
04030   5215          JMP LOOP          /REITERATE
04031   7346          STA CLL RTL
04032   7002          BSW               /SET AC=7577
04033   3235          DCA KNT           /SET COUNT TO ALLOW READING A
                                        /200 BYTE RECORD
04034   5201          JMP CRCCHK        /GO CHECK THE CRC
04035   7737   KNT,   7737              /ONES COMPLIMENT OF NUMBER OF
                                        /BYTES TO LOAD
04036   3557   PTR,   LOC-23            /MEMORY LOCATION TO BEGIN LOAD AT
04037   7730   M50,   -50               /CLA SPA SZL
               /THIS ROUTINE BINARY LOADS BINARY FILES INTO MEMORY.
               /IT BEGINS BY LOADING A RECORD OF SIZE 40,
               /THEN CONTINUES TO LOAD SUCCESSIVE RECORDS EACH OF SIZE
               /200.
               /THIS PROCESS CONTINUES UNTIL IT DESTROYS ITSELF.
               /[LOCATIONS 4000 AND 4001 ARE REPLACED BY JMP I(BIN)]
               /BY THE SECONDARY BOOTSTRAP.
               /THE FIRST MEMORY LOCATION BEFORE A NEW CASSETTE RECORD
               /IS READ IN IS LOADED WITH A RANDOM VALUE (173).
               /SUCCESSIVE WORDS ARE LOADED WITH THE 12-BIT QUANTITY.
               /100A+B, WHERE A AND B ARE SUCCESSIVE 6-BIT BYTES FROM
               /THE CASSETTE RECORD.
               /MEANINGLESS WORDS GET LOADED IF THE CASSETTE CONTAINS
               /8-BIT BYTES AS CAN (AND DOES) HAPPEN WHEN 'LOADING'
               /THE HEADER, AND WHEN 'LOADING' THE ORIGIN AT THE
               /BEGINNING OF THE RECORD.
```

Basically, the bootstrap file is a standard PDP-8 binary file except that origin settings are treated as data words. The ROM reads data from successive records (beginning with the header record) and treats this data as binary data (two successive bytes form one 12-bit word using the low order 6-bits of each byte). These binary data words are loaded into core into successive locations beginning with location 3557 in field 0. This location was chosen so that the random data in the file header block loads there and then the real data from the first 200-byte record begins loading into location 3602.

NOTE

> Each time a record gap is encountered, the next core location is loaded with an undetermined 12-bit word. Data in the next record resumes with the next core location.

This process continues until locations 4000 and 4001 are loaded. Location 4000 must be loaded with the starting address of the secondary bootstrap, location 4001 must be loaded with a 5600. (This number must be a multiple of 100).

It is strongly recommended that there be only one bootstrap file - namely the one used by CAPS-8 called C2B00T. If necessary, it can read in a terciary bootstrap to do further loading (see the OS/8 program called C3BOOT).

A.2 PDP-8 CASSETTE BINARY FORMAT

File consists of a sequence of entries.

Each entry consists of one or two 8-bit bytes as follows: (bit 8 is high order bit, bit 1 is low order):

data entry:	2-bytes
1st byte:	bits 8,7 must be 00 bits 6-1 are high order 6 bits of data word
2nd byte:	bits 8,7 must be 00 bits 6-1 are low order 6 bits of data word
origin entry:	2 bytes

1st byte:	bits 8,7 must be 01 bits 6-1 are high order 6 bits of new origin
2nd byte:	bits 8,7 must be 00 bits 6-1 are low order 6 bits of new origin
field entry:	1 byte bits 8,7 must be 11 bits 6-4 is new field setting bits 3-1 must be 000
trailer entry:	1 byte bits 8-1 must be 10 000 000

These entries may appear in any order except that trailer entries may only appear at the end of the file and there must be at least one byte of leader/trailer at the end. If the last entry before the final trailer is an origin entry then that represents the program's starting address.

Cassette binary format does not include a binary checksum. This was a design flaw.

First frame of trailer at end signifies logical end-of-file. Data after it is not specified.

A.3 PDP-11 CASSETTE FORMATS

1. Formatted Binary Format

   Records are word aligned, variable length and only the "text" section can cross block boundaries. Records are variable length and can cross block boundaries. Nulls are used as inter-record separators where necessary.

Byte 1	001
Byte 2	000
Byte 3	low order of (length of "DATA" in bytes)+4=[n]
Byte 4	high order of (length of "DATA" in bytes)+4=[n]
Byte 5	
Byte n	(last byte of DATA)

   Byte n+1 Checksum byte = - Byte

   (two's complement add with carry out ignored)

The format for the "data" information will vary from use to use.

NOTE

PDP-11 OBJ & LDA formats (types 20&22) both adhere to this formatted Binary Format, but differ in their interpretation of "DATA".

2. 11M TSK Format

Cannot be completely defined here.

*Since there will always be an 11M Task Builder Manual, a reference to it should suffice.

APPENDIX B

FORMAT OF CASSETTES FOR FIELD RELEASE

When a cassette is copied from one drive to another, there is no guarantee that the physical length of the data on the copy will be the same as the physical length of the original. Although data may fit on one cassette, when attempting to copy this data onto another cassette, one may run out of room on the second cassette. For this reason, we recommend that any cassettes intended for interchange should not be more than about 75% full.

More specifically, we recommend that a cassette intended for interchange should not contain more than 260000 (octal) bytes of data. When computing this number, each record gap written should be considered to be equivalent to 56 (octal) bytes of data and each file gap (including the initial one) should be considered equivalent to 454 (octal) bytes.

Cassettes intended to be copied by and distributed via the Digital Software Distribution Center must adhere to this recommendation.

NOTE

> Currently the SDC cannot copy a cassette bigger than this. (This affects submission of multi-volume cassette files to SDC).

APPENDIX C

NON-STANDARD FILE TYPES

Type	Description
2	paper tape image (non-ASCII)
3	core image format #1 one 36-bit computer word in 5 byte (wastes low order 4 bits of fifth byte)
4	core image format #2 one 12-bit computer word in 2 byte (only the low order 6 bits of each byte being used)
5	core image format #3 one 18-bit computer word in 3 byte
6	core image format #4 one 36-bit computer word in 6 byte (only the low order 6 bits of each byte being used)
7	core image format #5 one 16-bit computer word in 2 byte
10	PS/8 character packing (core image format #6), 3 bytes for two 12-bit words as shown below
11	core image format #7 two 36-bit words in 9 byte
12	core image format #8 four 18-bit words in 9 byte
13	bootstrap file
14	bad file