TBM File Format Reference & Utility Manual

Nicholas DeCicco

This work was performed under the auspices of the National Center for Atmospheric Research (NCAR) Earth Observing Laboratory (EOL) Summer Undergraduate Projects in Engineering Research (SUPER) program, which is managed by the University Corporation for Atmospheric Research (UCAR) and is funded by the National Science Foundation (NSF) (www.eol.ucar.edu).

Chapter 1

TBM File Format

1.1 Introduction

The Ampex TMS-4 Terabit Memory System ("TBM") was a tape storage system used at NCAR from approximately the mid 1970s through to the mid 1980s. Much of the GENPRO-I data from this time period on HPSS was formerly only available in TBM-formatted archives. This document describes the format of these archives, as well as the TBMconv software framework which may be used to extract the contents of TBM files.

1.2 TBM Format Overview

A TBM file begins with a SYSLBN block (see section 3.3) at offset 0. This block contains information such as:

- the machine used to generate the TBM file (e.g., the Cray 1),
- the data type (e.g., CDC DPC),
- the size of a "BK block," and
- offsets to specific locations in the file.

Of particular interest are the "BK block" size (expressed in multiples of 2048 60-bit words), the data type, and the offsets.

Bits 59–30 of word 29 ("offset to first file control pointer"; `firstFCPOff` in the API) of this SYSLBN block specify the offset, in 60-bit words, from the start of the file (offset 0) to the first file control pointer (see section 3.4). Each file control pointer is followed by a set of file history words (see section 3.5) and a set of block control pointers (see section 3.6). Each block control pointer is one 60-bit word in length, and there are a total of `numBKBlocks` of them.

For a TBM archive containing only one file, one should find that

$$\texttt{numBKBlocks} = \underbrace{\texttt{nextFCPOff}}_{\substack{\text{From first file}\\ \text{control pointer}\\ \text{located at}\\ \texttt{firstFCPOff}}} - (\texttt{firstFCPOff} + 9)$$

The length of the file is indicated by the values of `numBKBlocks` and `bk` in `SYSLBN`. The length of the file should be:

$$\text{file length}_{\text{8-bit bytes}} = \frac{(\texttt{numBKBlocks} + 1) \times \texttt{bk} \times 2048_{\text{60-bit words}} \times 60_{\text{bits per word}}}{8_{\text{bits per byte}}}.$$

The portions of a TBM file where actual data resides are broken into many *data records*; each data record is preceeded by a data buffer flags (see section 3.7) word, also known as a "record control word." Each data buffer flag contains an offset (relative to it) specifying the location of the next data buffer flag (`nextPtrOffset` in the `DataBufferFlags` struct).

File data (including metadata) begins at one TLIB/BK block into the file and ends with a Data Buffer Flag whose `isEOD` property is set to 1; padding follows afterwards as needed to make the total file size an integer multiple of TLIB/BK blocks long. Files are bookended by metadata consisting of alternating Data Buffer Flags and `VOL1`/`HDR1`/`HDR2` metadata

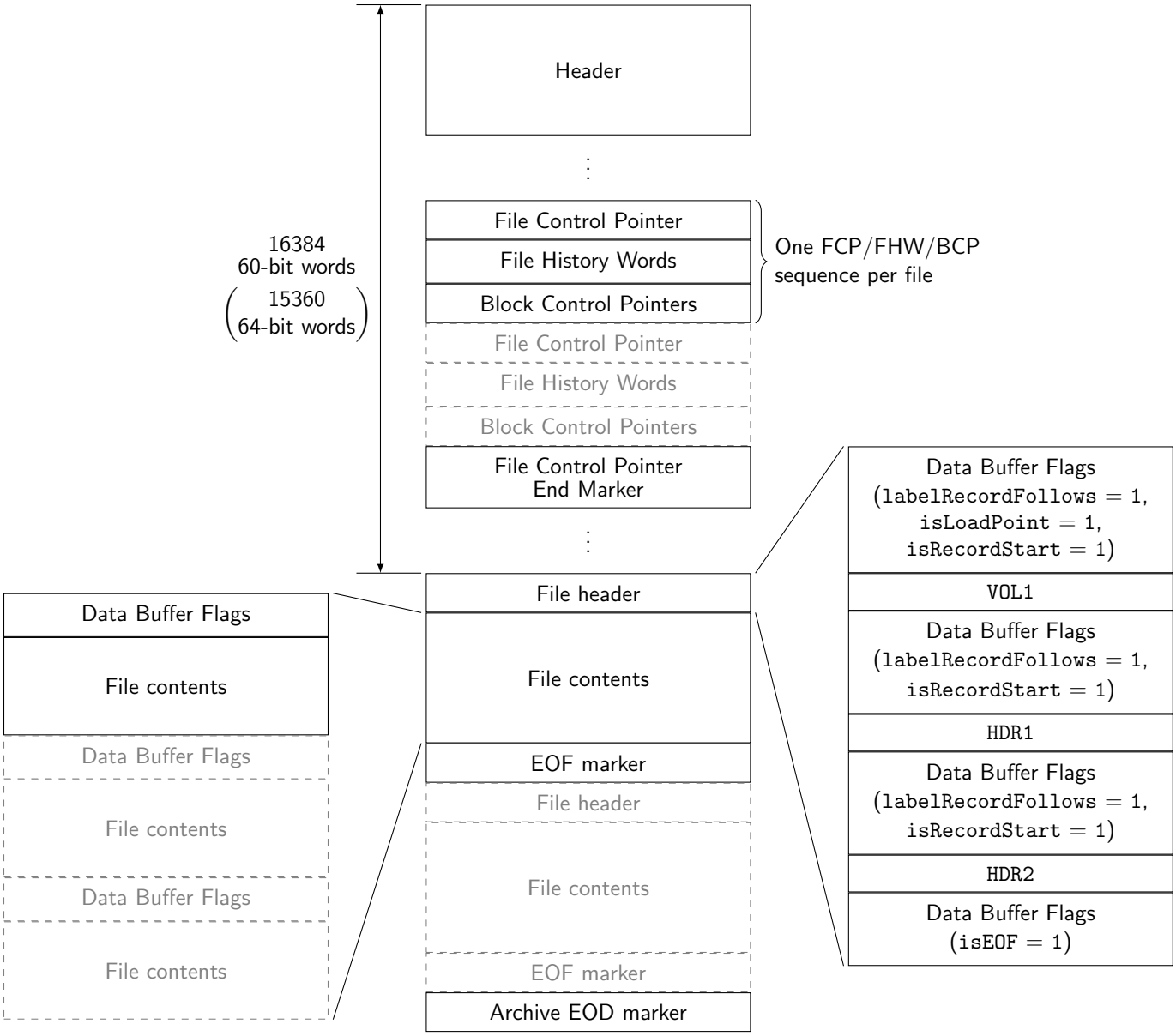


Figure 1.1: Illustration of the general TBM file layout.

at the beginning of each file and a `DataBufferFlags` structure at the end. The length of a TLIB block in terms of 60-bit words is

$$\text{total TBM file length}_{\text{60-bit words}} = 2048 \times \texttt{bk},$$

or in terms of 64-bit words,

$$\text{total TBM file length}_{\text{64-bit words}} = \frac{2048 \times \texttt{bk} \times 60}{64}.$$

For example, for the default `bk` value of 8,

$$\text{length}_{\text{60-bit words}} = 2048 \times 8 = 16384 \text{ 60-bit words},$$
$$\text{length}_{\text{64-bit words}} = \frac{2048 \times 8 \times 60}{64} = 15360 \text{ 64-bit words}.$$

A schematic diagram illustrating the most important elements of the TBM format is shown in Figure 1.1. An example from an actual file (NCAR/EOL dataset PHOENIX-78 file G50233) of offsets to data structures and some of their values illustrating the format is shown in Section 1.2.

Table 1.1: PHOENIX-78 Dataset file G50233 layout

0	SYSLBN	
51	FCP	
52–59	FHW	(dataSetID = 'NCARSYSTEMHD10001')
60–110	BCP	
111	FCP	
112–119	FHW	(dataSetID = 'NCARSYSTEMHD10002')
120–203	BCP	
204	FCP	
205–212	FHW	(dataSetID = 'NCARSYSTEMHD10003')
213–270	BCP	
271	FCP	
272–279	FHW	(dataSetID = 'NCARSYSTEMHD10004')
280–298	BCP	
299	FCP	(isEOF = 1)
⋮		
16384	DBF	(labelRecordFollows = 1, isLoadPoint = 1, isRecordStart = 1)
16385	VOL1	(volSerialName1 = 'G50233', tbmVolSerial = 'TL0110')
16393	DBF	(labelRecordFollows = 1, isRecordStart = 1)
16394	HDR1	(dataSetID = 'NCARSYSTEMHD10001')
16402	DBF	(labelRecordFollows = 1, isRecordStart = 1)
16403	HDR2	
16411	DBF	(isEOF = 1, isRecordStart = 1)
16412	DBF	(isRecordStart = 1)
⋮		
840063	DBF	(isEOF = 1)
840064	DBF	(labelRecordFollows = 1)
840065	EOF1	(dataSetID = 'NCARSYSTEMHD10001')
840073	DBF	(isEOF = 1, endLabelGroup = 1)
840074	DBF	(labelRecordFollows = 1)
840075	HDR1	(dataSetID = 'NCARSYSTEMHD10002')
840083	DBF	(labelRecordFollows = 1)
840084	HDR2	
840092	DBF	(isEOF = 1, endLabelGroup = 1)
840093	DBF	
⋮		
2200434	DBF	(isEOF = 1)
2200435	DBF	(labelRecordFollows = 1)
2200436	EOF1	(dataSetID = 'NCARSYSTEMHD10002', blockCount = 3317)
2200444	DBF	(isEOF = 1, endLabelGroup = 1)
2200445	DBF	(labelRecordFollows = 1)
2200446	HDR1	(dataSetID = 'NCARSYSTEMHD10003')
2200454	DBF	(labelRecordFollows = 1)
2200455	HDR2	
2200463	DBF	(isEOF = 1, endLabelGroup = 1)
2200464	DBF	
⋮		
3139325	DBF	(isEOF = 1)
3139326	DBF	(labelRecordFollows = 1)
3139327	EOF1	(dataSetID = 'NCARSYSTEMHD10003', blockCount = 2289)
3139335	DBF	(isEOF = 1, endLabelGroup = 1)
3139336	DBF	(labelRecordFollows = 1)
3139337	HDR1	(dataSetID = 'NCARSYSTEMHD10004')
3139345	DBF	(labelRecordFollows = 1)

3139346	HDR2	
3139354	DBF	(isEOF = 1, endLabelGroup = 1)
3139355	DBF	
⋮		
3431236	DBF	(isEOF)
3431237	DBF	(labelRecordFollows = 1)
3431238	EOF1	(dataSetID = 'NCARSYSTEMHD10004', blockCount = 711)
3431246	DBF	(isEOF = 1, endLabelGroup = 1)
3431247	DBF	(nextPtrOffset = 0, prevPtrOffset = 1, isEOD = 1, isRecordStart = 1)

1.2.1 Minimum Requirements for Reading a TBM Archive

For a TBM archive containing only one file, the only information from the first TLIB/BK block (containing the SYSLBN data structure as well as the file control pointers, file history words, and buffer control pointers) that is strictly necessary for reading a TBM archive is the "BK block" size (specified in the `bk` element of the `SYSLBN_Data` structure in the API). With this information, the file contents can be extracted by stripping data buffer flags from the second TLIB/BK block through to the end of the file (denoted by a data buffer flags word with `isEOD` set to 1).

For a TBM archive containing multiple files, the same is sufficient, except that one must make note of the start and end of separate files within the archive, denoted by special data buffer flag / `VOL1` / `HDR1` / `HDR2` sequences described previously.

Chapter 2

Using `tbmexplore`

`tbmexplore` is a command-line utility for interactively exploring/investigating the contents TBM files. `tbmexplore` permits dumping contents of a TBM file at arbitrary offsets, optionally decoding the

```
$ tbmexplore G51452
Enter an offset:
```

We enter 0 and press enter, and are next prompted with:

```
How would you like to display the data?

         1) 6-bit DPC
         2) Data Buffer Flags (aka ‘‘Record Control Word")
         3) Block Control Pointer
         4) File Control Pointer
         5) File History Word
         6) SYSLBN
         7) NotQuiteSYSLBN
         8) VOL1
         9) HDR1
        10) HDR2
        11) 20-bit integers
        12) 60-bit integers

Enter a choice [1-12]:
```

Choosing 6, we see

```
 ==== SYSLBN ====
offset = 0 (bits) 0+0 (8-bit bytes) 0+0 (60-bit words)
machineType         =        0 (CDC 7600)
density             =        0 (200 BPI)
dataType            =        0 (BCD as DPC)
numTracks           =        1
bk                  =        8
numBKBlocks         =      308
labelBufLen         =     1024
 ===== VOL1 =====
vol1                = "VOL1" (0x58F31C)
volSerialName1      = "G51452"
acc                 = " " (45; unlimited access)
acntNum             = "41113306" (0x01F71C1C79E6E1)
sciNum              = "5&" (0x837)
tbmVolSerial        = "TL0483" (0x50C6DF8DE)
sysLevelCode        = " " (0x2D)
 ===== HDR1 =====
hdr1                = "HDR1" (0x20449C)
dataSetID           = "NCARSYSTEMHD10001"
volSerialName2      = "G51452"
fileSecNum          = "0001"
fileSeqNum          = "0001"
generationNum       = "0001"
versionNum          = "00"
creationDate        = " 82320"
```

```
expDate             = " 83320"
accChar             = " " (0x2D)
blockCount          = "000000"
sysCode             = "NCAR   SYSTEM"
 ===== HDR2 =====
hdr2                = "HDR2" (0x20449D)
hdr2label           = "                                                                                "
 ==== SYSLBN ====
fileCtrlPtrOff      =      0 ("      ")
blkCtrlPtrOff       =      0 ("      ")
firstFCPOff         =     51 ("     %")
ctrlCardOpenOff     =     35 ("     8")
openMergeAreaOff    =     35 ("     8")
curCtrlCardOpenOff  =     35 ("     8")
fcpToBlkCtrlOff     =     10 ("     J")
```

Chapter 3

TBM Format and API Reference

3.1 API Introduction

tbmconv includes a header file (tbm.hpp) containing data structures useful for reading TBM files, as well as a set of routines for printing the contents of these structures to the terminal (contained in tbm.cpp). These data structures map to structures specified in the reference as illustrated in table 3.1.

Table 3.1

Data structure	API data structure names	
	Raw data	DPC
SYSLBN	SYSLBN_Data	SYSLBN_Text
VOL1	VOL1_Data	VOL1_Text
HDR1	HDR1_Data	HDR1_Text
HDR2	HDR2_Data	HDR2_Text

The functions provided in tbm.cpp are:

```
void read_syslbn(uint8_t const*const inBuf, SYSLBN_Text *const text,
                 SYSLBN_Data *const data, const size_t offset);
void read_fileHistoryWord(uint8_t const*const inBuf,
                          FileHistoryWord_Text *const text,
                          FileHistoryWord_Data *const data,
                          const size_t offset);
void read_dataBufferFlags(uint8_t const*const inBuf,
                          DataBufferFlags *const dbf,
                          const size_t offset);
void read_fileControlPointer(uint8_t const*const inBuf,
                             FileControlPointer *const fcp,
                             const size_t offset);
void read_vol1(uint8_t const*const inBuf,
               VOL1_Text *const text,
               VOL1_Data *const data,
               const size_t offset);
void read_hdr1(uint8_t const*const inBuf,
               HDR1_Text *const text,
               HDR1_Data *const data,
               const size_t offset);
void read_hdr2(uint8_t const*const inBuf,
               HDR2_Text *const text,
               HDR2_Data *const data,
               const size_t offset);

void print_vol1(VOL1_Text const*const text, VOL1_Data const*const data,
                const size_t offset);
void print_hdr1(HDR1_Text const*const text, HDR1_Data const*const data,
                const size_t offset);
void print_hdr2(HDR2_Text const*const text, HDR2_Data const*const data,
                const size_t offset);
```

```
void print_syslbn(SYSLBN_Text const*const text, SYSLBN_Data const*const data,
                  const size_t offset);
void print_fileControlPtr(FileControlPointer const*const fcp,
                          const size_t offset, const int printHorizontal,
                          const int printHeader);
void print_fileHistoryWord(FileHistoryWord_Text const*const fhw_text,
                           FileHistoryWord_Data const*const fhw_data,
                           const size_t offset);
void print_blockControlPointer(BlockControlPointer const*const bcp,
                               const size_t offset, const int printHorizontal,
                               const int printHeader);
void print_dataBufferFlags(DataBufferFlags const*const dbf,
                           const size_t offset, const int printHorizontal,
                           const int printHeader);
void print_offset(const size_t offset);
```

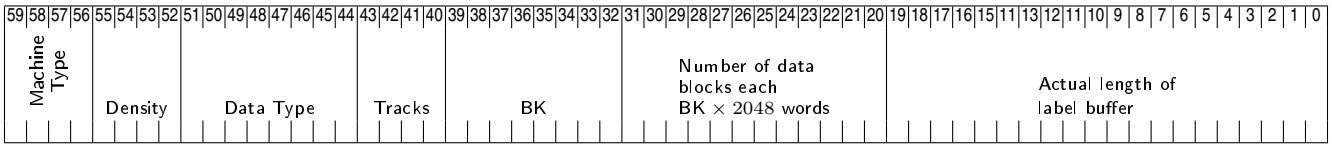
3.2 Accessibility critera

The accessability critera characters are a character which indicate “any restrictions on who may have access to the information in this file. A space means unlimited access; any other character means special handling, in a manner to be defined."

3.3 SYSLBN

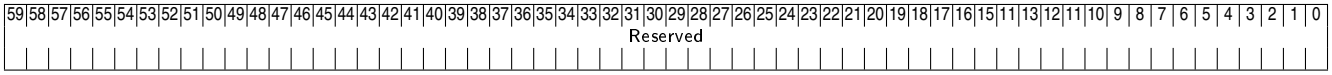
A SYSLBN block appears at the start of every TBM file, at offset 0. The SYSLBN block could be thought of as being comprised of a header (words 0 through 3), a VOL1 label (words 4–11), a HDR1 label (words 12–19), a HDR2 label (words 20–27), and finally pointers to various locations in the file (words 28–30).

3.3.1 SYSLBN Word 0



Length	Bits	Type	API	Description
4	59–56	Binary	machineType	The machine type. 0 7600 1 Cray-1 2 Front end
4	55–52	Binary	density	The original density. 0 200 bpi 1 556 bpi 2 800 bpi 3 1600 bpi
8	51–44	Binary	dataType	The data type. 0 BCD as DPC 1 Binary bit-serial 2 BCD, no conversion from 7-channel stage-in 3 ASCII 4 EBCDIC
4	43–40	Binary	numTracks	The original number of tape tracks 0 7-track 1 9-track
8	39–32	Binary	bk	The value of BK from a TLIB card on the 7600
12	31–20	Binary	numBKBlocks	The number of BK length Data Blocks in this volume
20	19–0	Binary	labelBufLen	Actual length of the label (SYSLBN) buffer

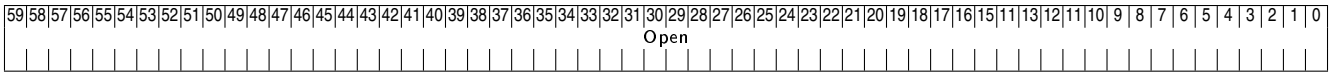
3.3.2 SYSLBN Word 1



Length	Bits	Type	API	Description
60	59–0	—	—	Reserved

3.3.3 SYSLBN Word 2

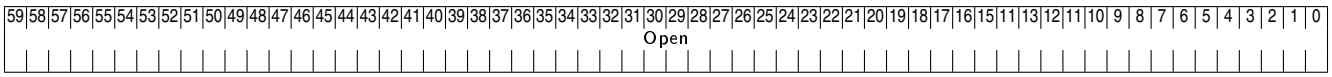
This word is unused and should be set to zero.



Length	Bits	Type	API	Description
60	59–0	—	—	Open

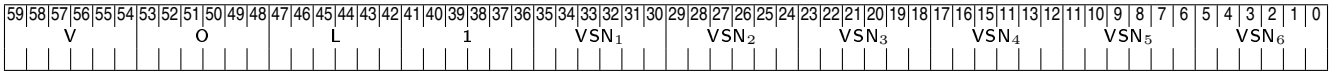
3.3.4 SYSLBN Word 3

This word is unused and should be set to zero.



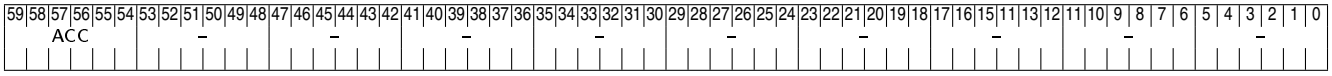
Length	Bits	Type	API	Description
60	59–0	—	—	Open

3.3.5 SYSLBN Word 4 (VOL1 Word 0)



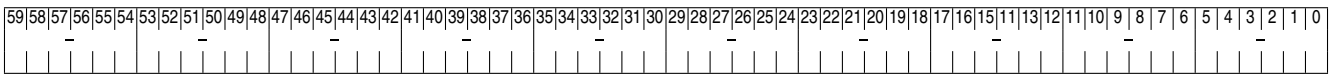
Length	Bits	Type	API	Description
24	59–36	DPC	`vol1`	VOL1 characters for a Volume One label and the start of the Volume One label
36	35–0	DPC	`volSerialName1`	Six Volume Serial Name (VSN) characters

3.3.6 SYSLBN Word 5 (VOL1 Word 1)



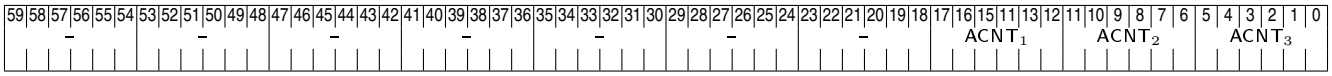
Length	Bits	Type	API	Description
6	59–54	DPC	`acc`	Volume One label accessability criteria (see section 3.2)
54	53–0	—	—	Blank fill characters

3.3.7 SYSLBN Word 6 (VOL1 Word 2)



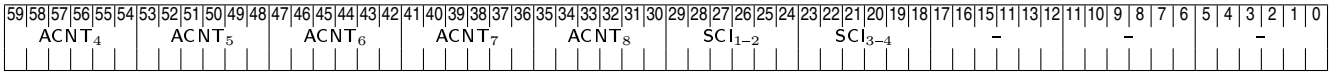
Length	Bits	Type	API	Description
60	59–0	—	—	Blank fill characters

3.3.8 SYSLBN Word 7 (VOL1 Word 3)



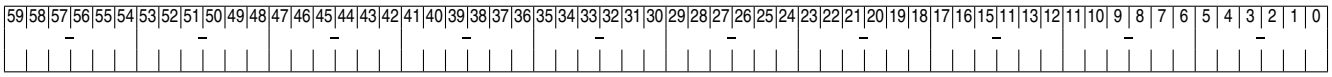
Length	Bits	Type	API	Description
42	59–18	—	—	Blank fill characters
18	17–0	DPC	acntNum_1_3	Characters 1–3 of the *JOB card accounting number

3.3.9 SYSLBN Word 8 (VOL1 Word 4)



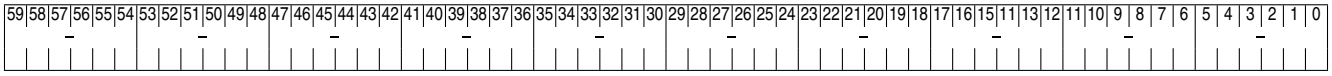
Length	Bits	Type	API	Description
30	59–30	DPC	acntNum_4_8	Characters 4–8 of the *JOB card accounting number
12	29–18	?	sciNum_1_4	The scientist number from the *JOB card
18	17–0	—	—	Blank fill characters

3.3.10 SYSLBN Word 9 (VOL1 Word 5)



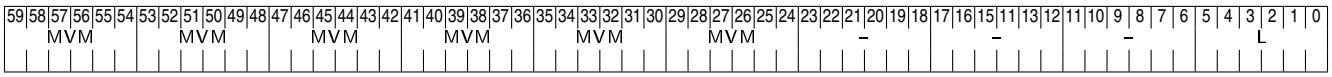
Length	Bits	Type	API	Description
60	59–0	—	—	Blank fill characters

3.3.11 SYSLBN Word 10 (VOL1 Word 6)



Length	Bits	Type	API	Description
60	59–0	—	—	Blank fill characters

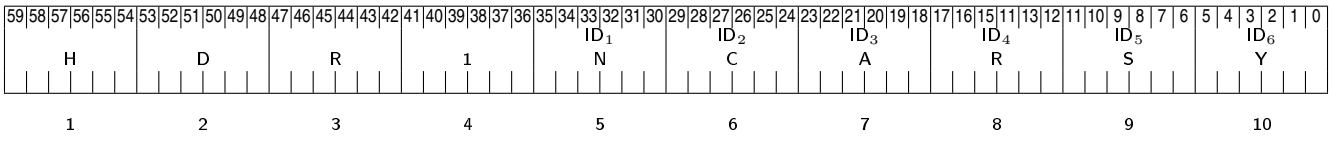
3.3.12 SYSLBN Word 11 (VOL1 Word 7)



Length	Bits	Type	API	Description
36	59–24	DPC	tbmVolSerial	TBM Volume Serial Name
18	23–6	—	—	Blank fill characters
6	5–0	DPC	sysLevelCode	System level code

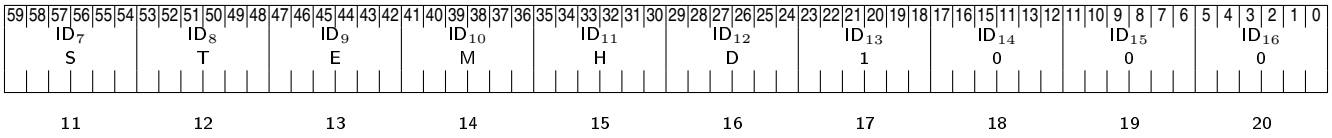
3.3.13 SYSLBN Word 12 (HDR1 Word 0)

This word marks the beginning of the "header one label," which is 80 6-bit characters long.



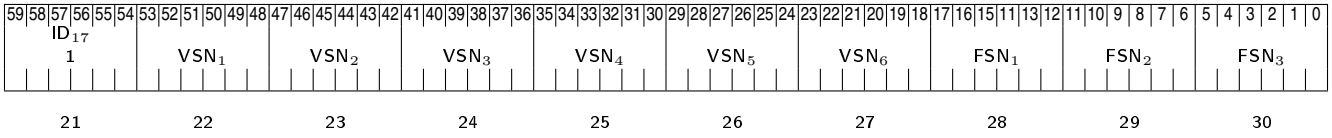
Length	Bits	Type	API	Description
24	59–36	DPC	hdr1	HDR1 characters for a Header One label and the start of the first Header One label
36	35–0	DPC	dataSetID_1_6	Data Set Identifier (DSI) characters 1–6

3.3.14 SYSLBN Word 13 (HDR1 Word 1)



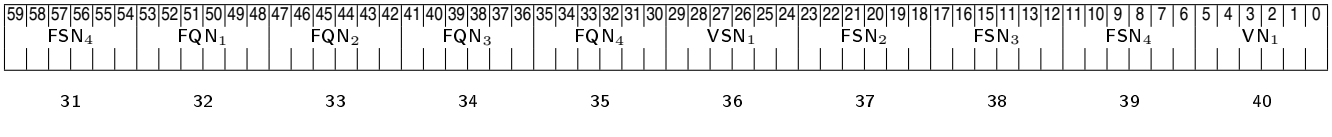
Length	Bits	Type	API	Description
60	59–0	DPC	dataSetID_7_12 dataSetID_13_16	Data Set Identifier (DSI) characters 7–16

3.3.15 SYSLBN Word 14 (HDR1 Word 2)



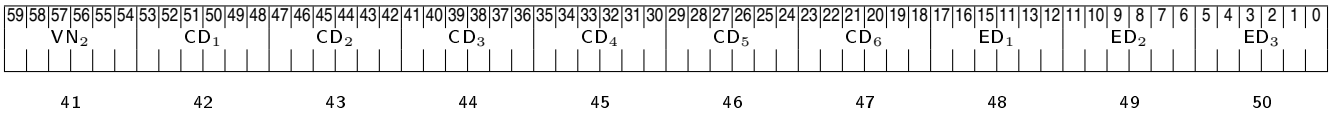
Length	Bits	Type	API	Description
6	59–54	DPC	dataSetID_17	Data Set Identifier (DSI) character 17
36	54–18	DPC	volSerialName2	Volume Serial Name characters 1–6, equal to the ones in word 4, bits 59–36
18	17–0	DPC	fileSecNum_1_3	File section number characters 1–3

3.3.16 SYSLBN Word 15 (HDR1 Word 3)



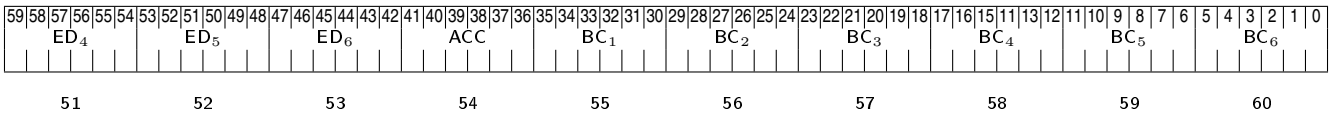
Length	Bits	Type	API	Description
6	59–54	DPC	fileSecNum_4	File section number character 4
24	53–30	DPC	fileSeqNum	File sequence number characters 1–4
24	29–6	DPC	generationNum	Generation number characters 1–4
6	5–0	DPC	versionNum_1	Version number character 1

3.3.17 SYSLBN Word 16 (HDR1 Word 4)



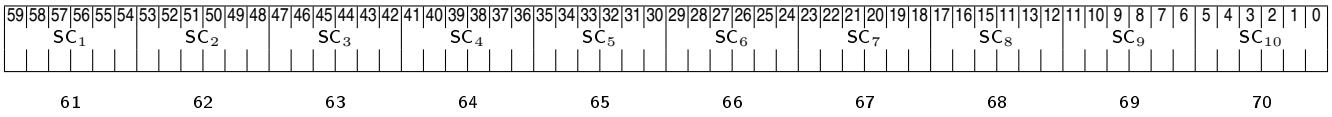
Length	Bits	Type	API	Description
6	59–54	DPC	versionNum_2	Version number character 2
36	53–18	DPC	creationDate	Creation date characters 1–6
18	17–0	DPC	expDate_1_3	Expiration date characters 1–3

3.3.18 SYSLBN Word 17 (HDR1 Word 5)



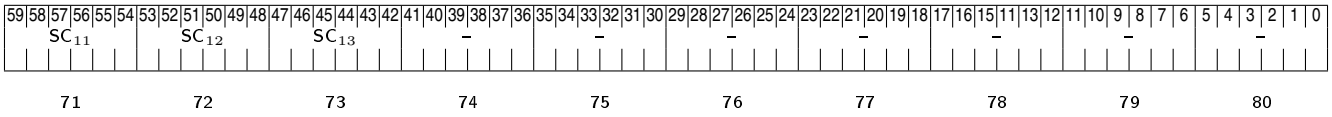
Length	Bits	Type	API	Description
18	59–42	DPC	expDate_4_6	Expiration date characters 4–6
6	41–36	DPC	accChar	The HDR1 accessability criteria character (see section 3.2)
36	35–0	DPC	blockCount	The block count characters

3.3.19 SYSLBN Word 18 (HDR1 Word 6)



Length	Bits	Type	API	Description
60	59–42	DPC	sysCode_1_10	System code characters 1–10

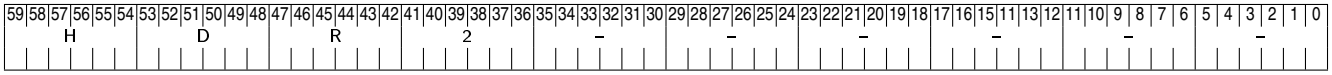
3.3.20 SYSLBN Word 19 (HDR1 Word 7)



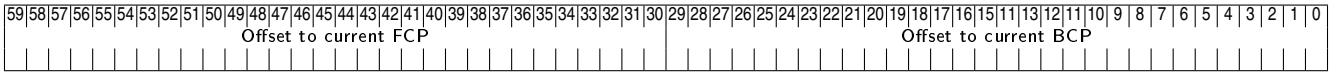
Length	Bits	Type	API	Description
18	59–42	DPC	sysCode_11_13	System code characters 11–13
42	41–0	—	—	Spaces

3.3.21 SYSLBN Words 20–27 (HDR2 Words 0–7)

These words contain the HDR2 label.

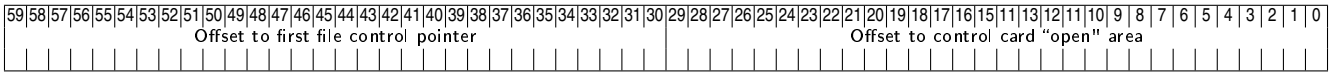


3.3.22 SYSLBN Word 28



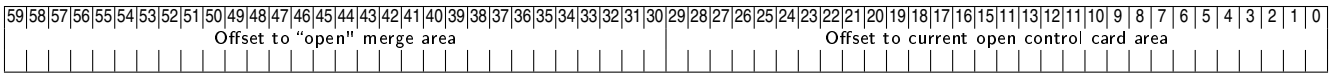
Length	Bits	Type	API	Description
30	59–30	Binary	fileCtrlPtrOff	Offset to current file control pointer
30	29–0	Binary	blkCtrlPtrOff	Offset to current block control pointer from SYSMSI

3.3.23 SYSLBN Word 29



Length	Bits	Type	API	Description
30	59–30	Binary	firstFCPOff	Offset to first file control pointer
30	29–0	Binary	ctrlCardOpenOff	Offset to control card “open" area

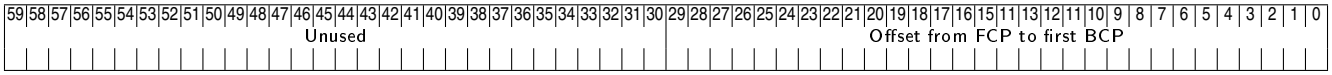
3.3.24 SYSLBN Word 30



Length	Bits	Type	API	Description
30	59–30	Binary	openMergeAreaOff	Offset to “open" merge area
30	29–0	Binary	curCtrlCardOpenOff	Offset to current open control card area

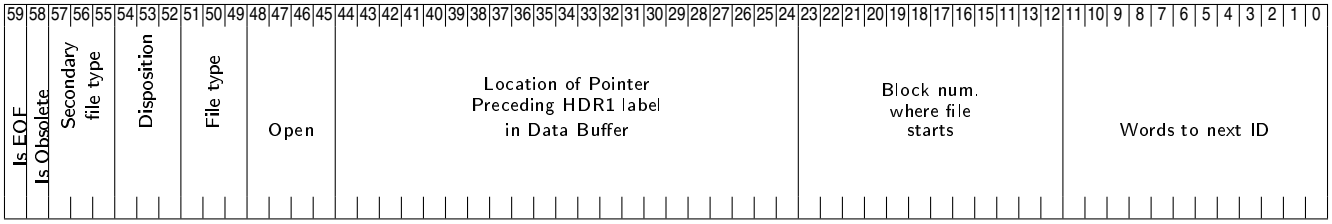
3.3.25 SYSLBN Word 31

Note that `fcpToBlkCtrlOff` appears to be off by one; it has a value of 10 when it should be 9.



Length	Bits	Type	API	Description
30	59–30	—	—	Unused
30	29–0	Binary	`fcpToBlkCtrlOff`	Offset from File Control Pointer to first Block Control Pointer

3.4 File Control Pointers

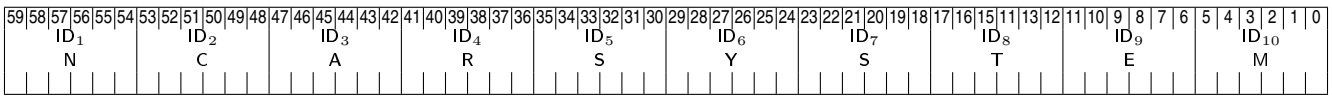


Length	Bits	Type	API	Description
1	59	Binary	`isEOF`	If set, this FCP indicates EOF; no other bits used.
1	58	Binary	`isObsolete`	If set, file is obsolete
3	57–55	Binary	`secondaryFileType`	Secondary file type 1 old 2 new 4 scatch
3	54-52	Binary	`fileDisposition`	File disposition 0 keep 1 delete at close 2 delete at termination
3	51-49	Binary	`fileType`	File type 0 undefined 1 sequential access 2 direct access 3 mixed access
4	48–45	—	—	Open
21	44–24	Binary	`bufferPtrOffset`	Location of buffer pointer preceding the HDR1 label for this file
12	23–12	Binary	`dataBlkNum`	Data block number where the file starts
12	11–0	Binary	`nextFCPOff`	Words to next file control pointer

3.5 File History Words

The file histoy words immediately follow every file control pointer, hence why the numbering of these words starts at 1.

3.5.1 File History Word 1

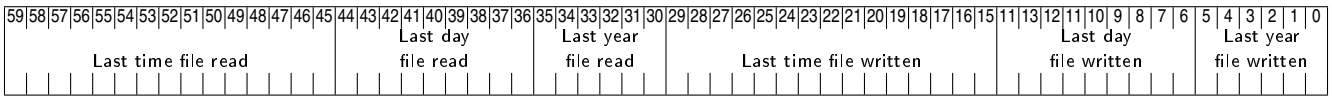


Length	Bits	Type	API	Description
60	59–0	DPC	`dataSetID_1_10`	Characters 1–10 of the Data Set Identifier (DSI) from the current HDR1 label

3.5.2 File History Word 2

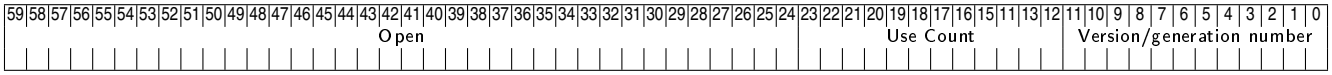
Length	Bits	Type	API	Description
42	59–24	DPC	dataSetID_13_17	Characters 11–17 of the Data Set Identifier (DSI) from the current HDR1 label
18	17–0	—	—	Spaces

3.5.3 File History Word 3



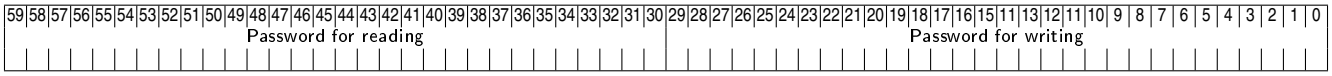
Length	Bits	Type	API	Description
15	59–45	Binary	lastReadTime	Last time the file was read
9	44–36	Binary	lastReadDay	Last day of year file was read
6	35–30	Binary	lastReadYear	Last year file was read ($0 = 1976$)
15	29–15	Binary	lastWriteTime	Last time the file was written
9	14–6	Binary	lastWriteDay	Last day of year file was written
6	5–0	Binary	lastWriteYear	Last year file was written ($0 = 1976$)

3.5.4 File History Word 4



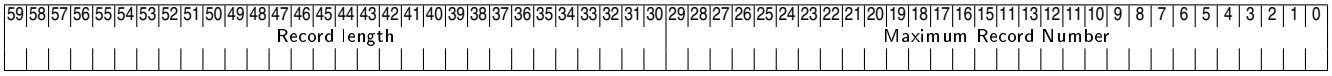
Length	Bits	Type	API	Description
36	59–24	—	—	Open
12	23–12	Binary	useCount	Use count; i.e., the number of times the file was referenced. Needed for PLIB.
12	11–0	Binary	versionNum	Version number/generation number

3.5.5 File History Word 5



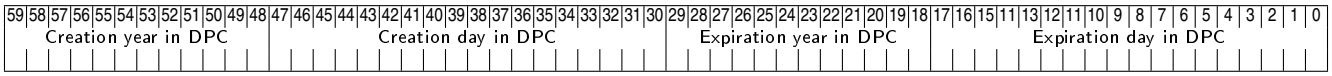
Length	Bits	Type	API	Description
30	59–30	DPC	readPasswd	Password for reading
30	29–0	DPC	writePasswd	Password for writing

3.5.6 File History Word 6



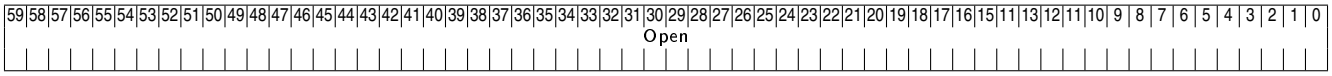
Length	Bits	Type	API	Description
30	59–30	Binary	recordLen	Record length
30	29–0	Binary	maxRecordNum	Maximum record number

3.5.7 File History Word 7



Length	Bits	Type	API	Description
12	59–48	DPC	`creationYear`	Creation year
18	47–30	DPC	`creationDay`	Creation day
12	29–18	DPC	`expirationYear`	Expiration year
18	17–0	DPC	`expirationDay`	Expiration day

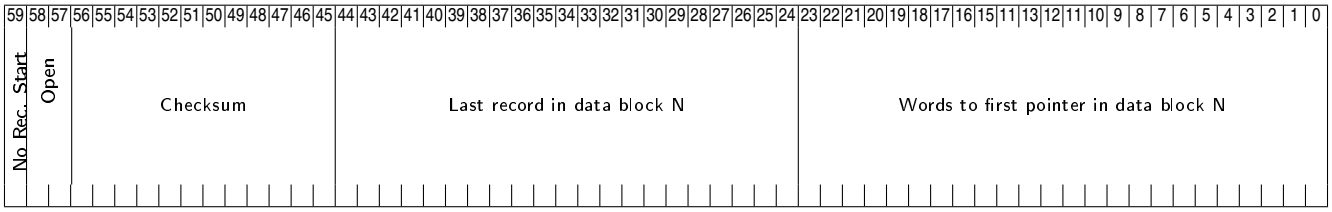
3.5.8 File History Word 8



Length	Bits	Type	API	Description
60	59–0	—	—	Open

3.6 Block Control Pointer (BCP)

Sets of Block Control Pointers follow each set of File History Words, which in turn folow each File Control Pointer.



Length	Bits	Type	API	Description
1	59	Binary	`noRecordStartsHere`	If set, no record starts in this block.
2	58–57	—	—	Open
12	56–45	Binary	`checksum`	Checksum[1]
21	44–24	Binary	`lastRecord`	Last record starting in data block N
24	23–0	Binary	`wordsToFirstPtr`	Words to first pointer in data block N

3.6.1 Further Remarks

Representation in NCAR Technical Note

Note that the way the block control pointer is depicted in the NCAR Technical Note is rather confusing; the technical note shows what at first appear to be three distinct "block control pointers," one which is "word 9", another which is word "$+9+M-1$", and another which is word "$+9+M$". Really what the technical note is trying to convey is that there are M block control pointers per file; the first block control pointer starts at an offset $+9$ after the first file control pointer; and immediately after the last block control pointer ($+9+M$) is another file control pointer. If there are multiple files in a TBM archive, there will be multiple sets of block control pointers.

The `lastRecord` field

`lastRecord` is the one-based index of the last record to appear in the $\texttt{bk} \times 2048$-sized data block corresponding to a given buffer control pointer. For example, using `tbmexplore` to view the first three buffer control pointers shows that the "last record in data block N" for these are 35, 72, and 108.

[1] Not mentioned in the NCAR Tech Note, but can be seen referenced on line 96 of the `as.s` CAL source file for `tbm2cos`.

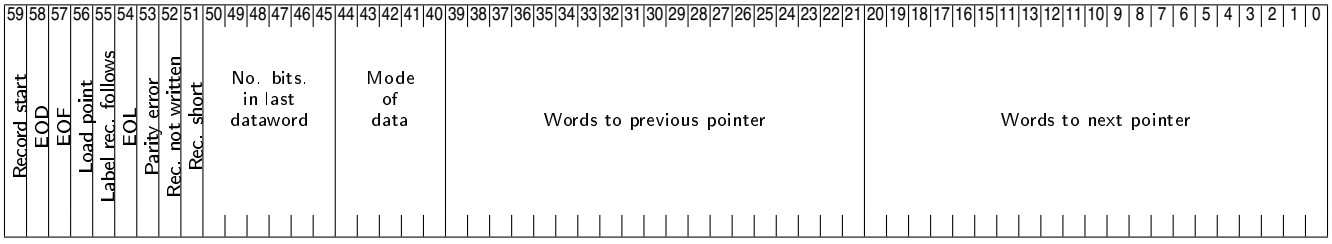
offset	no record starts here	checksum	last record in data block N	words to first pointer in data block N
60	0	3496	35	220
61	0	2517	72	365
62	0	2696	108	520

Using `tbmexplore` to dump the offsets of the data buffer flags containing actual data in the same file, then taking those offsets mod 16384 to show where the offsets cross block boundaries, we can see that the indices of the records at the end of each block correspond to the aforementioned values:

Index	Offset to Data Buffer Flag (aka "Record Control Word")	Offset mod 16384
1	16412	28
2	17743	1359
3	18190	1806
⋮		
33	31600	15216
34	32047	15663
35	32494	16110
36	32941	173
37	33388	620
38	33835	1067
⋮		
70	48139	15371
71	48586	15818
72	49033	16265
73	49480	328
74	49927	775
75	50374	1222
⋮		
106	64231	15079
107	64678	15526
108	65125	15973
109	65572	36
110	66019	483
111	66466	930

3.7 Data Buffer Flags

The Data Buffer Flags are interspersed within actual file data and bookend the start and end of each file. The first Data Buffer Flag in a TBM archive is located at the beginning of the second `TLIB/bk` block. All successive Data Buffer Flags can be found by moving forward in the file by the amount specified in each `nextPtrOffset`. The Data Buffer Flags are also referred to as the "record control word" or the "data buffer field."



Length	Bits	Type	API	Description
1	59	Binary	isRecordStart	Start of record
1	58	Binary	isEOD	End of data in this volume
1	57	Binary	isEOF	End of file
1	56	Binary	isLoadPoint	Load point for tape simulation
1	55	Binary	labelRecordFollows	Label record follows
1	54	Binary	endLabelGroup	End of label group file mark
1	53	Binary	sourceRecordHasParityError	Source record had a parity error
1	52	Binary	recordNotWritten	Record not written (direct access volume)
1	51	Binary	recordIsShorter	Record shorter than words to next pointer-1, length is in bits 0-21 of first data word
6	50–45	Binary	numBits	Number of bits in last data word controlled by this pointer
5	44–40	Binary	recordDataMode	Mode of data in this record (see Data Type bits 51–44 of first word of SYSLBN)
19	39–21	Binary	prevPtrOffset	Words to previous buffer pointer
21	20–0	Binary	nextPtrOffset	Words to next buffer pointer

3.8 First End-Of-File Label (EOF1)

All of the characters in EOF1 correspond directly to those in HDR1, but only characters 5–54 and 61–80 of EOF1 will have the (exact) same values as those of HDR1 in a given file; characters 55-60 contain the block count, which changes between HDR1 and EOF1. Characters 1–4 change from HDR1 to EOF1, so only the first word (word 0) of EOF1 is shown.

For comparison, consider the HDR1 and EOF1 at the offsets shown in the table below from the CODE-II file G51452:

Field	HDR1 Value	EOF1 Value
Offset[2]	16394	5050071
hdr1/eof1	"HDR1" (0x20449C)	"EOF1" (0x14F19C)
dataSetID	"NCARSYSTEMHD10001"	"NCARSYSTEMHD10001"
volSerialName2	"G51452"	"G51452"
fileSecNum	"0001"	"0001"
fileSeqNum	"0001"	"0001"
generationNum	"0001"	"0001"
versionNum	"00"	"00"
creationDate	" 82320"	" 82320"
expDate	" 83320"	" 83320"
accChar	" " (0x2D)	" " (0x2D)
blockCount	"000000"	"011259"
sysCode	"NCAR SYSTEM"	"NCAR SYSTEM"

Observe that the block count changes from 0 to 11259; in this file, there are 11259 records between HDR1 and EOF1.

3.8.1 EOF1 Word 0

Length	Bits	Type	API	Description
18	59–42	DPC		The string "EOF1".
42	41–0	DPC		Same as in HDR1.

offset	offset to next FCP	data blk number	buffer ptr offset	file type	file disposition	2nd type	is obsolete?	is EOF?
51	349	0	201	0	0	0	0	0

[2] "Offset" here is not a field in HDR1 or EOF1, but rather the offset in the file to the start of each of these sequences.

400	10	339	8486	0	0	0	0	0
410	0	0	0	0	0	0	0	1