Series 1100

Data Structures

Programmer Reference

Page Status Summary

Section	Pages	Update
Cover/Disclaimer		
PSS	1	
Preface	1 - 2	
Contents	1 - 8	
Section 1	1 - 2	
Section 2	1 - 11	
Section 3	1 - 7	
Section 4	1 - 51	
Section 5	1 - 15	
Section 6	1 - 26	
Section 7	1 - 40	
Section 8	1 - 17	
Section 9	1 - 58	
Glossary	1 - 7	
User Comment Sheet		

Section	Pages	Update

Preface

This manual contains information on the data structures used in the SPERRY Series 1100 Operating System. It is primarily a reference and not an instructional tool.

Data Structures is, at this publication, roughly equivalent to EXEC level 38R2. The EXEC-related information will not be updated with each EXEC release. Therefore, you should not assume that the data structures described here necessarily correspond to those in use at your site.

These are the software levels which create the corresponding data structures. They are listed by section as they appear in this manual.

Section		Software	Level
2.	Program File Format	EXEC (PFP) SYSLIB (BSP$)	38R2 75R1
3.	FURPUR File Formats	FURPUR	28R3
4.	System Data Format (SDF)	EXEC (Symbiont) SYSLIB (SIR$, SOR) PCIOS	38R2 75R1 4R1
5.	Relocatable Binary Format (RBF)	SYSLIB (ROR) Collector MASM	75R1 31R1 4R1
6.	Absolute Element Format	Collector	31R1
7.	Master File Directory (MFD)		
8.	Tape Labelling System	EXEC	38R2
9.	Log Entry File Formats		

The fact that this information is being made available does not imply that Sperry encourages or supports modifications of software using the information supplied. It is supplied for convenience only and will be used at your own risk.

The data structures described are used in many different software products; more information can be found in the following Sperry manuals:

- UP-4144.2 *Series 1100 Executive System, EXEC, Programmer Reference*
- UP-7876 *Series 1100 FORTRAN V Library, Programmer Reference*
- UP-8204 *Series 1100 FORTRAN V Common Bank I/O Library, Programmer Reference*
- UP-8448 *Series 1100 System Generation, EXEC, User Guide*
- UP-8453 *Series 1100 Meta-Assembler (MASM), Programmer Reference*
- UP-8724 *Series 1100 FURPUR, Programmer Reference*
- UP-8728 *Series 1100 System Relocatable Library and Common Bank (SYSLIB), Programmer Reference*
- UP-8730 *Series 1100 System Utilities, Programmer Reference*
- UP-9047 *Series 1100 Assembly Instruction Mnemonics (AIM), Supplementary Reference*

Check the *Series 1100 Summary of Current Documentation,* UP-7893, for the current or applicable software version of these manuals.

Contents

User Comment Sheet

Figures

Tables

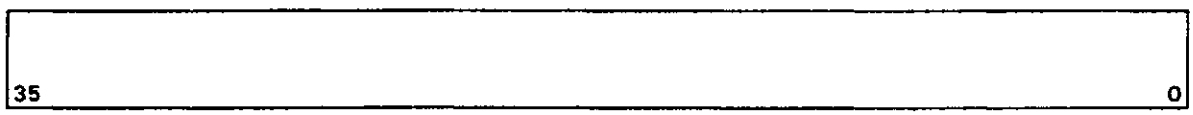
1. Introduction

This manual contains information on the data structures used in the SPERRY Series 1100 Operating System. It is intended for programmers who need this information to work with the Operating System. Since this manual is primarily a reference and not an instructional tool, many of the sections consist of tables and figures diagramming formats, entries, and relationships of the data structures.

Sections 2 through 6 cover formats for program files, FURPUR files, System Data (SDF), Relocatable Binary (RB), and absolute elements. Section 7 describes the Master File Directory (MFD). Tape labelling is detailed in Section 8.

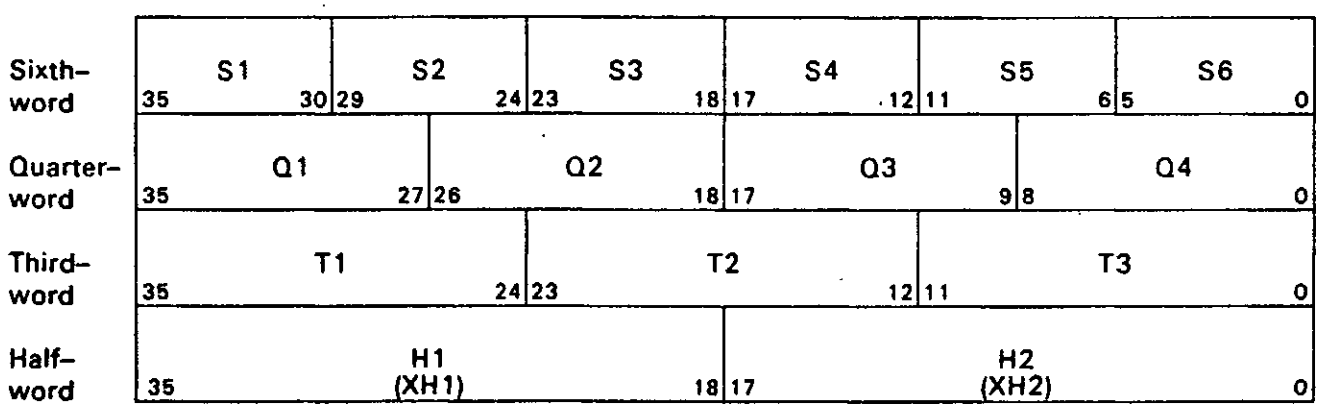
Again, this manual is intended to be a reference. Necessary information can often be found quickly by consulting the list of figures and tables in the table of contents.

In this document, the 36 bits in the Series 1100 computer word are numbered from right to left. For example:



(The mnemonic W is used in ambiguous situations to indicate whole word references.)

To reference partial words, the following mnemonics are used:



The Series 1100 Assembler mnemonics are used whenever machine instructions are discussed. For more information on these mnemonics, reference the MASM, Programmer Reference, UP-8453 (see Preface). The Assembler mnemonics for partial word transfers and registers are defined in the AIM Supplementary Reference, UP-9047 (see Preface).

The mnemonics U and XU indicate immediate data references (operand taken directly from address portion of instruction rather than from the main storage referenced by that address). The mnemonics XH1, XH2, and XU indicate that the 18-bit value is to be loaded with sign-extension to 36 bits (i.e., if bit 17 is a one, bits 35-18 are set to one).

Control registers are referenced by the following mnemonics:

- A0, A1,...,A15 Accumulators (A0 - A3 can also be used as index registers).
- A15+1, A15+2 Additional accumulators for double- or triple-register instructions.
- X1, X2,..., X11 Index registers.
- R0, R1, R2,..., R15 R-registers.

Activities may use one of two sets of control registers, the major set (all control registers as given above), or the minor set (registers X8 through X11, A0 through A5, and R1, R2, and R3).

Subroutines frequently use a subset of the minor set, called volatile registers (i.e., data left in these registers may not be saved). This subset includes registers X11, A0 through A5, and R1, R2, and R3

The dollar sign ($) is generally used in system-defined external symbols, procedure names, and filenames; to avoid duplication, you should not use this character. The $ generally occurs as the last character of a symbol, excepting procedure names in which it is usually the second character.

In packet and table formats, parameters in regular type indicate information that must be supplied by the programmer; parameters in italics indicate information that the system returns. Brackets ([ ]) are used to indicate optional parameters.

The symbol Δ is used to indicate a blank character.

In control statement, Executive Request, and procedure call formats, capital letters represent themselves and must be coded as shown. Lowercase letters represent variables which must be coded as directed in the text.

Numbers are represented as in assembler syntax; that is, a leading zero specifies octal.

2. Program File Format

A program file is a partitioned random access file consisting of a group of elements residing on mass storage. A program file may contain symbolic, relocatable, omnibus, or absolute elements or any combination of these types. It may be either a temporary or a cataloged file. Since the elements are named, they may be manipulated on an individual basis.

It must be emphasized that while the program file is logically structured as shown in Figure 2-1, the elements that physically make up the file are not necessarily contiguous. Links are automatically generated by the Executive to logically structure the file as a continuous entity.

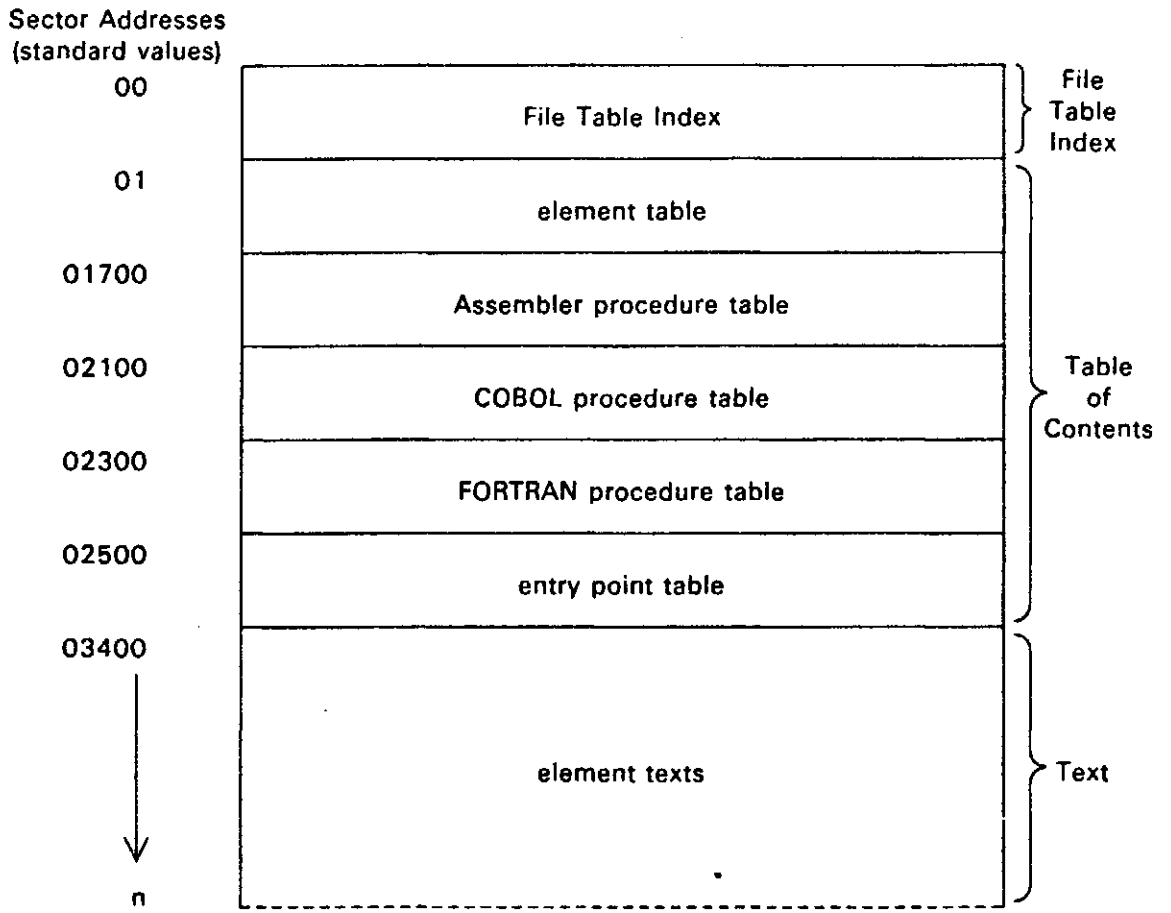


Figure 2-1. Program File Format

The program file (Figure 2-1) has three major sections:

1. File Table Index – Contains pointers and links (relative to the beginning of the file) to the tables that make up the file's table of contents (see Table 2-1).

2. Table of Contents – Provides pointers to the elements (element table), procedures (procedure tables), and entry points for relocatable binary elements (entry point table). These tables are illustrated and described in the following paragraphs. Each table, except for the element table which always starts at sector 1, begins at its system standard sector, or at a greater sector if a table has extended beyond the standard starting sector, or at a lesser sector if the file has been packed (@PACK) with the M option. The system standard starting sectors are defined in the element ERU$ in SYS$*RLIB$. In Figure 2-2, the maximum number of entries shown for each table is based on each table starting at its system standard sector address. The maximum number of Element Table entries is 5000 if there are no other table entries.

	Tracks	Maximum Number of Entries (standard values)
Element Table	15	2671
Assembler Procedure Table	2	861
COBOL Procedure Table	2	430
FORTRAN Procedure Table	2	861
Entry Point Table	7	3101

Figure 2-2. Program File Table Sizes

3. Text – The data which constitutes the text of an element resides in a contiguous set of sectors. The format of this data is dependent on the element type.

Table 2-1. File Table Index Format

00	*	*	P F * *
01	next sector available for writing text		
02	*run-id*		
03	scratch area (value will not be retained)		starting sector address of element table on mass storage
04	length of element table in words		unused
05	pointer table length – 1	element table item size	unused
06–10	Assembler procedure table information (similar to Words 03–05)		

(continued)

Table 2-1. File Table Index Format (continued)

011–013	COBOL procedure table information (similar to Words 03–05)		
014–016	FORTRAN procedure table information (similar to Words 03–05)		
017–021	entry point table information (similar to Words 03–05)		
022	size of largest preamble	unused	new relocatable element
023	sequence number of latest absolute element in file		
024	starting sector of element text		
025	0		
≈	≈		
031	0		
032	0		
033	TIME$ value		

The File Table Index is initially zero-filled.

Word 00 — This word, when it contains "**PF**" (in Fieldata), identifies the file as a program file.

Word 01 — Next available sector at which an element may be written.

Word 02 — *Run-id.*

Words 03–05 — Element table information.

	Change Indicator - Set nonzero to indicate that the table segment presently in the buffer has been modified and should be written back to mass storage.
Words 06-010	Assembler procedure table information.
Words 011-012	COBOL procedure table information.
Words 014-016	FORTRAN procedure table information.
Words 017-021	Entry point table information; destroyed when relocatable element is inserted in the table of contents created by @PREP.
Word 022 (S6)	Nonzero - Relocatable element added to program file. Zero - Absolute element added to program file. Allows automatic collection of the elements in the file TPF$ for @XQT when new relocatable elements have been added.
Word 023	Sequence number of last absolute element added to the program file. This will be zero if there are no absolute elements in the program file or if the last absolute element added has been deleted.
Word 033	TIME$.

2.1. Element Table

The element table (see Figure 2-3) contains the Fieldata element and version name of each element, its type (symbolic, relocatable, omnibus, or absolute), and pointers to its text within the program file. It also provides information concerning:

- the size of the element
- the time and date the element was created
- the sequence number specifies the order in which the element texts are entered in the file; the sequence number of the element in the file is used for linking entries within the element table
- the address of the text

The element table has the first 140 words (5 sectors) reserved as a pointer table and control word. The first 139 words are used to hold pointers that start chains. The elements that belong to a particular chain are determined by dividing the sum of the words containing the element name by the length of the pointer table (139) and using the remainder as an index into the pointer table. A total of 278 chains may exist because each word contains two pointers; the pointer in H1 is used if the quotient was odd, and the pointer in H2 is used if it was even. The control word contains the element table item size in H1 and the total number of entries in H2.

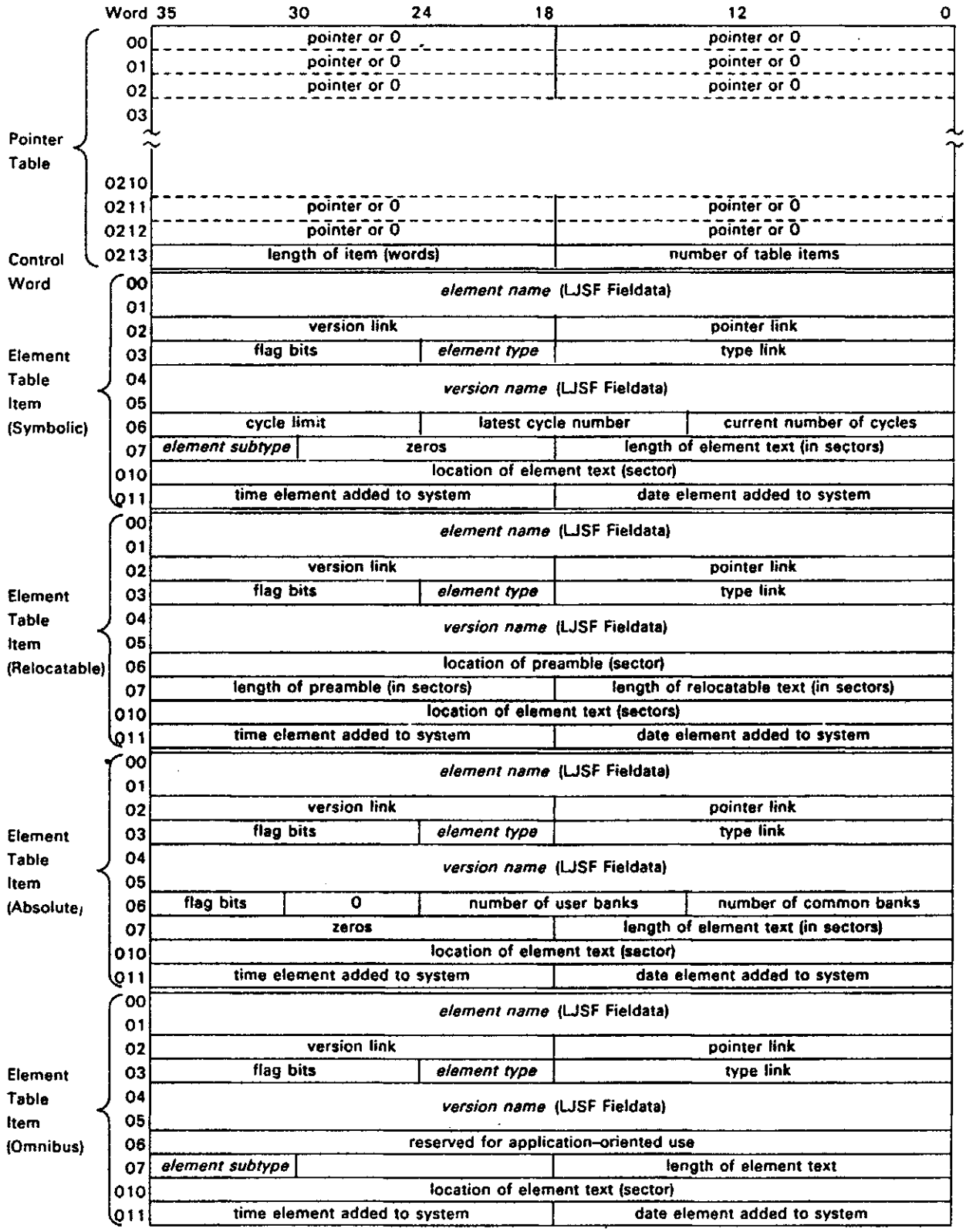


Figure 2–3. Element Table Format

Elements are pointed to and linked by sequence number. The sequence number for the first element added is 1, the second 2, through 5000.

The element table items follow the pointer table and appear in the order the elements were added to the program file.

When two or more elements need to be pointed to from the same pointer table half-word, element chains are formed and linked with pointers within the element table entries themselves.

The common fields of the various element table items are:

Word 2

version link — Sequence number of another element item with the same element name and type, but a different version, or the same version (but deleted).

pointer link — Sequence number of another element item with the same hash code (same pointer table index), but a different name.

Word 3

flag bits

Bit 35 - Marked for deletion

31 - CTS/HVTS flag (SYM). Indicator that an element is structured in CTS/HVTS format.

Bit 31 = 1:

(1) Only CTS/HVTS can set the bit when a new element is created.

(2) If only the entire element is copied (on a block-by-block basis, with all control word information intact), then the flag bit should be set in the output element if it was set in the input element.

Bit 31=0:

(1) If an element is modified, or not copied as outlined in (2) above (i.e., only the symbolic lines), then the flag bit must be cleared.

(2) An element created by any processor other than CTS/HVTS should have the CTS flag bit entry set to 0.

30 - Arithmetic fault non-interrupt mode (ABS, REL, SYM)
29 - Arithmetic fault compatibility mode (ABS, REL, SYM)
28 - ASCII Code (SYM) or real-time load (ABS)
27 - Block count in Bank Load Table is for 64-word blocks (ABS)
26 - Third-word sensitive (ABS, REL, SYM)
25 - Quarter-word sensitive (ABS, REL, SYM)
24 - Marked in error (ABS, REL, SYM)

element type — The element types are:

01 SYM - Symbolic
02 ASMP - Assembler procedure
03 COBP - COBOL procedure
04 FORP - FORTRAN procedure
05 REL - Relocatable

06	ABS	- Absolute
07	OMN	- Omnibus
08		- reserved
09	SKR	- Symbolic Keyed Random
010	SKR	- segment

Types 02, 03, and 04 are also symbolic elements in structure and content. Their corresponding element table items are identical to the symbolic element table item (type 01). Elements of types 01-04 will match (i.e., be indistinguishable) on a search of the Table of Contents (TOC) if they have the same element name and version. This means a SYM and FORP element cannot have the same element and version name and both be non-deleted.

type link — Sequence number of another element item with the same element name, but with a different type.

Word 6 (ABS) — Flag bits:

Bit 35	Always set
33	Requires two PSRs due to initial and utility basing
32	Suppresses zero fill
31	No starting address for program

Word 7 (SYM,OMN)

element subtype — The SYSLIB Programmer Reference, UP-8728 (see Preface) contains the presently defined omnibus and symbolic element subtypes (SSTYP$).

Word 9: — Time and date element was created (in reversed ER TDATE$ format):

time in seconds from midnight	month	day	year (modulo 1964)

2.2. Procedure Tables

The procedure table has an entry for each procedure (entered in the file by the @PDP control statement, described in System Utilities Programmer Reference UP-8730 (see Preface). Each entry consists of:

- the procedure name,
- a link to the element in which it appears, and
- the procedure's relative word location within the file.

Each procedure table, like the element table, contains a 139-word pointer table and a control word.

All procedure table items contain an element link, a pointer link, and the location of the procedure within the file as well as the procedure name. The pointer link is the sequence number of another item in the same procedure table with a different name, but the same hash code. It may be zero. The element link is the sequence number of the element table item associated with the procedure element containing the procedure. The location of the procedure is relative to the beginning of the file.

Bit 35 of Word 3 of each procedure table item is the delete flag. If set, that procedure has been deleted, either by a delete request or by the insertion of a new procedure with the identical name (replacement of the old procedure).

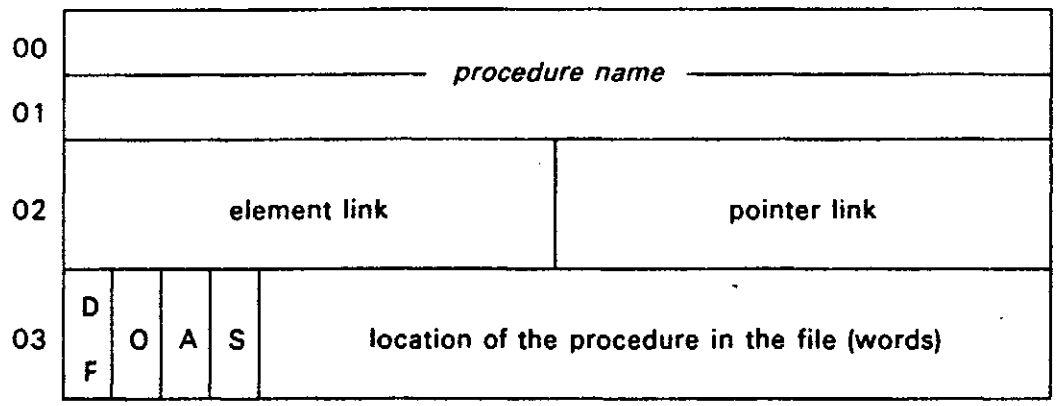
Bit 34 of Word 3 is the continuation indicator that, when set, shows that a second four words were necessary to contain the COBOL procedure name.

Bit 33 of Word 3 indicates that the procedure images are in ASCII.

Bit 32 of Word 3 indicates that the images contain sequence numbers in columns 73–80, which should be ignored.

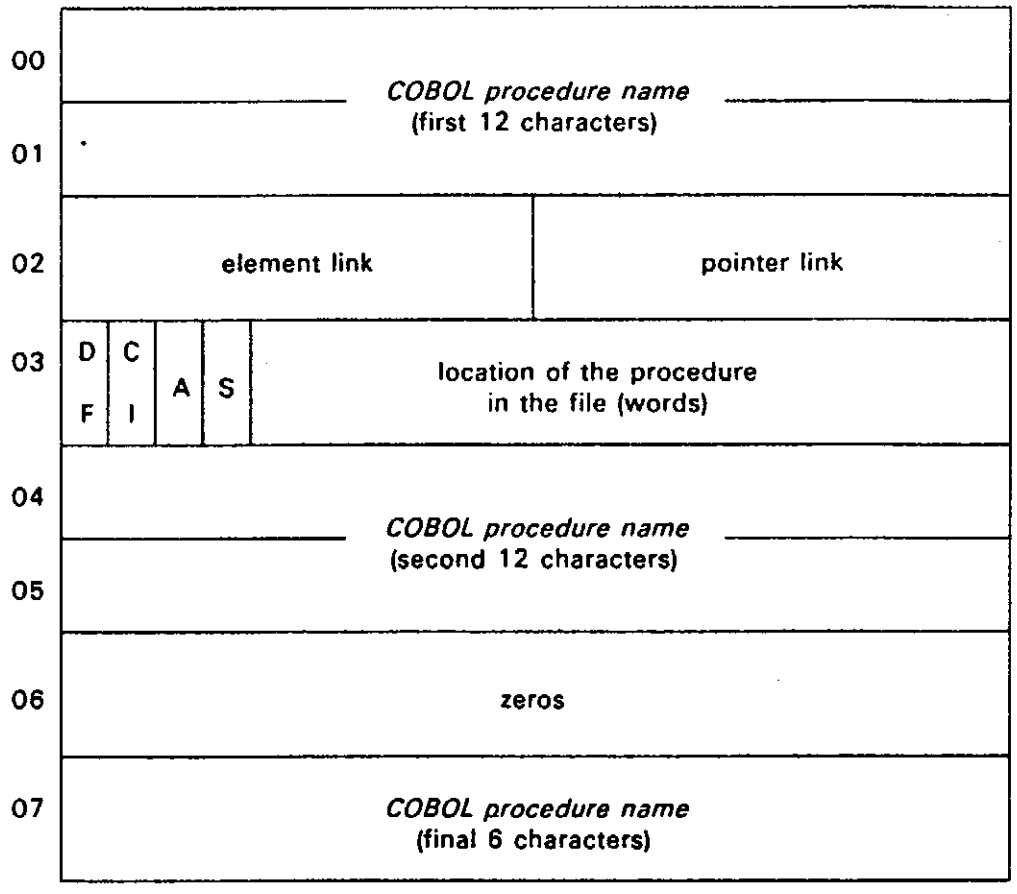
The Assembler and FORTRAN procedure entries are as shown in Table 2–2.

Table 2–2. Assembler or FORTRAN Procedure Table Item



The COBOL procedure table item (Table 2–3) is either four or eight words long. If the procedure name is 12 alphanumeric characters or less, the item will be four words long. If the procedure name exceeds 12 characters (the system limits the name to 30), a second four words are used to complete the item. Bit 34 of Word 3, when set, indicates that a second four words were necessary to contain the COBOL procedure name. Unused spaces in the COBOL procedure name field will be Fieldata space filled.

Table 2-3. COBOL Procedure Table Item



The first four words are always present.

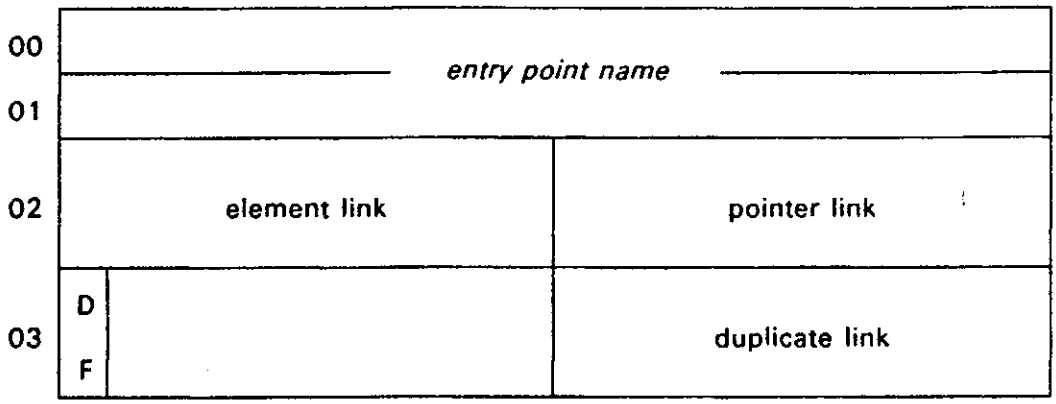
The second four words are present only if *COBOL procedure name* exceeds 12 characters and bit 34, Word 3 = 1.

2.3. Entry Point Table

The entry point table is the set of all entry point names and the link from each name to the relocatable element in which it occurs. The user must request the generation of this information using the @PREP control statement described in FURPUR Programmer Reference UP-8724 (see Preface); it is not done automatically by the Executive.

The entry point table contains a 139-word pointer table and a control word (like the element table). The entry point table item is shown in Table 2-4.

Table 2-4. Entry Point Table Item



The element link is the sequence number of the element table item associated with the relocatable element containing the entry point.

The pointer link is the sequence number of another item in the entry point table with a different name, but the same hash code.

The duplicate link is the sequence number of another item in the entry point table with the same name, but with a different element link.

Bit 35 of Word 3, if set, indicates the entry point is deleted.

2.4. Program File Maintenance

Program files are created and processed by the language processors, FURPUR, ELT, CULL, and other processors. There are Executive service functions and relocatable library routines available for processing program files. See the Program File Basic Service Package (BSP$) in the SYSLIB Programmer Reference, UP-8728 (see Preface).

3. FURPUR File Formats

3.1. Basic File Formats

Figure 3-1 illustrates the relationships of files to each other. The exact formats have been simplified for clarity. The control statements illustrated are control statements that transfer data between the indicated file types.

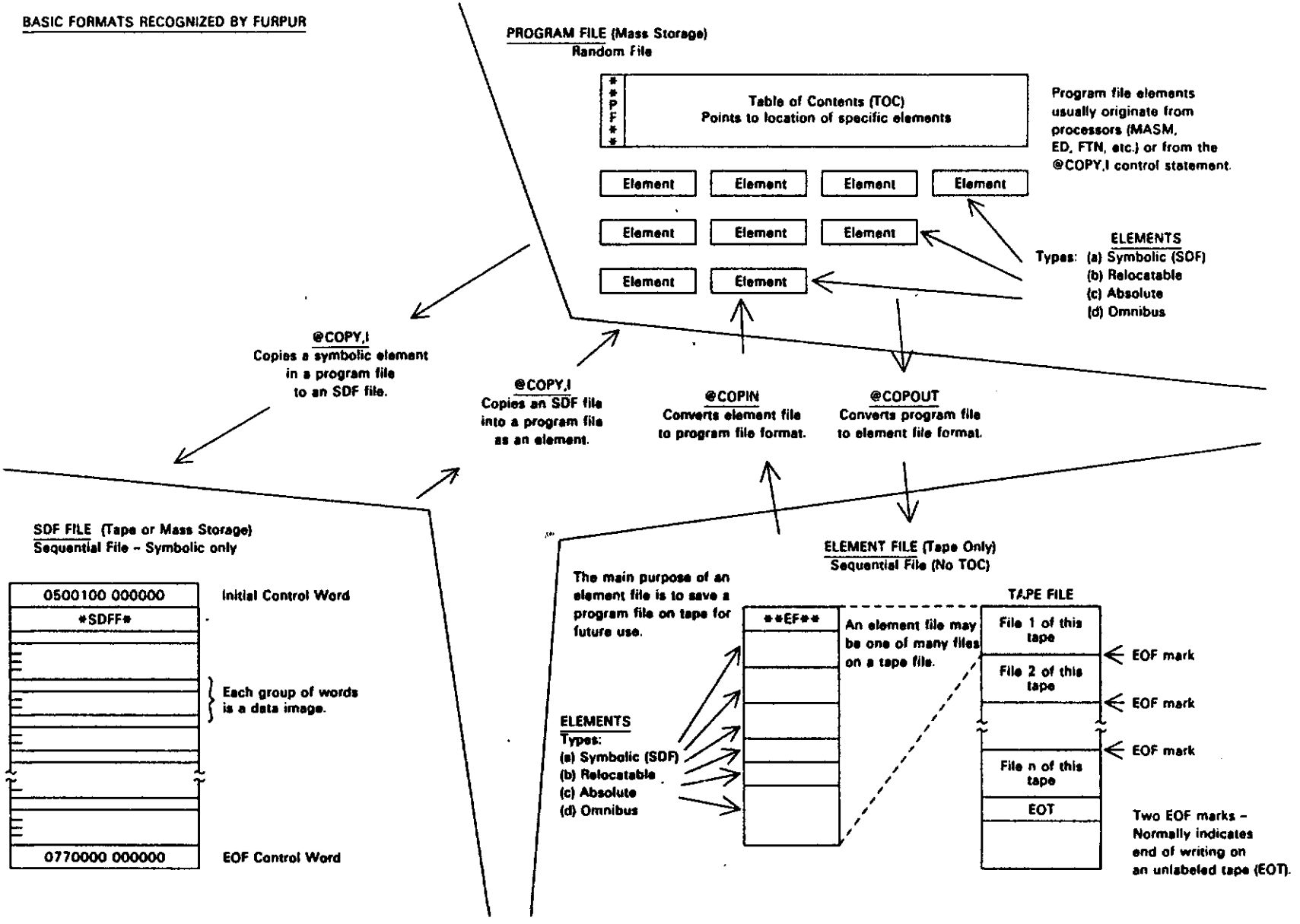


Figure 3-1. FURPUR Control Statements Used to Alter File Formats

3.2. @COPY,G File Format

The @COPY,G command copies sector-formatted files from mass storage to tape or from tape to mass storage without regard to the format of the contents.

A 28-word label block is written to tape prior to copying a file. The block's contents give details of the copied file and the time when the copy was made. See Table 3-1.

Table 3-1. FURPUR @COPY,G Format

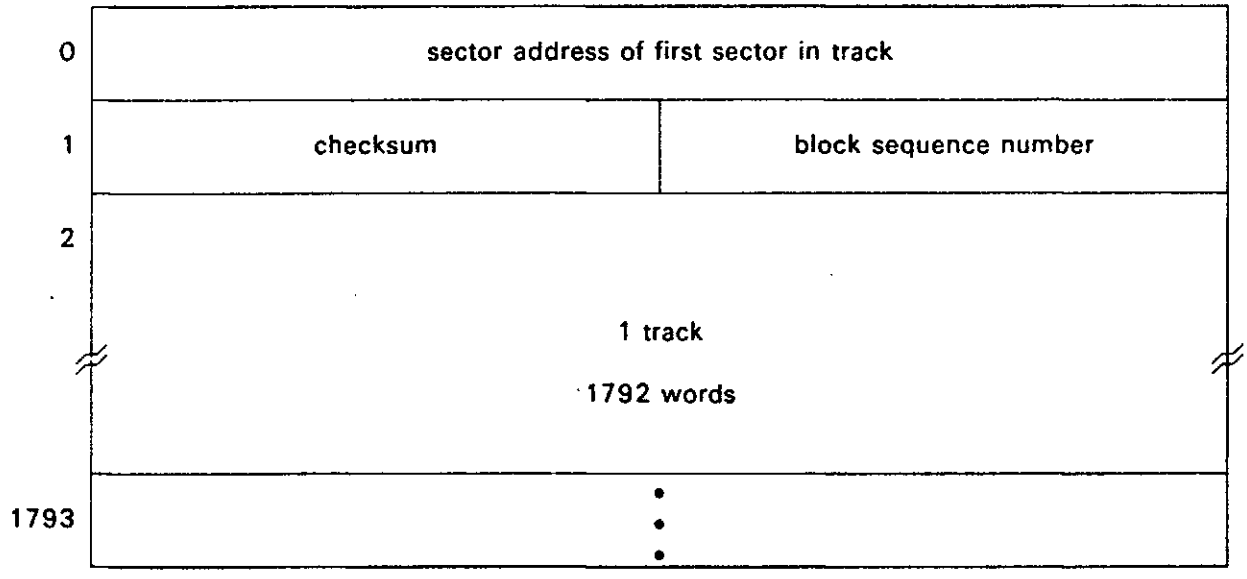
00	COPYG (LJSF Fieldata)
01	BLKSEQ (Fieldata)
02	*qualifier* (LJSF Fieldata)
03	
04	*filename* (LJSF Fieldata)
05	
06	absolute F-cycle (RJSF Fieldata)
07	date mmddyy (Fieldata)
010	time hhmmss (Fieldata)
011	equipment code
012	highest track written
≠	0 ≠
034	0

Subsequent blocks are 1794 words in length and consist of a sector-formatted mass storage track (1792 words) preceded by a 2-word header containing the track address and a checksum and block sequence number. (See Figure 3-2.)

The checksum is calculated by adding the 1792 words of the track together and then adding the two halves of the resulting value to form an 18-bit result.

Format:

Figure 3-2. Header and Track Block



The end-of-file and end-of-reel conditions conform to the same formats described for the @COPOUT control statement (see 3.4).

3.3. Element File Format

An element file is produced from a program file by using a @COPOUT control statement (see the FURPUR Programmer Reference, UP-8724 (see Preface). It is a sequential file, found only on magnetic tape, and it may consist of a series of symbolic, relocatable, absolute, and omnibus elements. It may be temporary or a cataloged file.

The elements (see Figure 3-3) are written in sequential order on the tape. Each element (see Figure 3-4) contains a 28-word element label block and the element text. The element label block consists of the 10-word program file element table item followed by 18 words of zero and contains the following information:

- element file identifier
- element name, version, type, and size
- time and date the element was added to the file

The remainder of the element consists of 224-word blocks of the text of the element. This information is identical to the element text in the program file from which the element file was created. The only difference is that the element text is blocked into 224-word blocks; the last block is padded to force a 224-word block if the text does not occupy an exact multiple of 224 words.

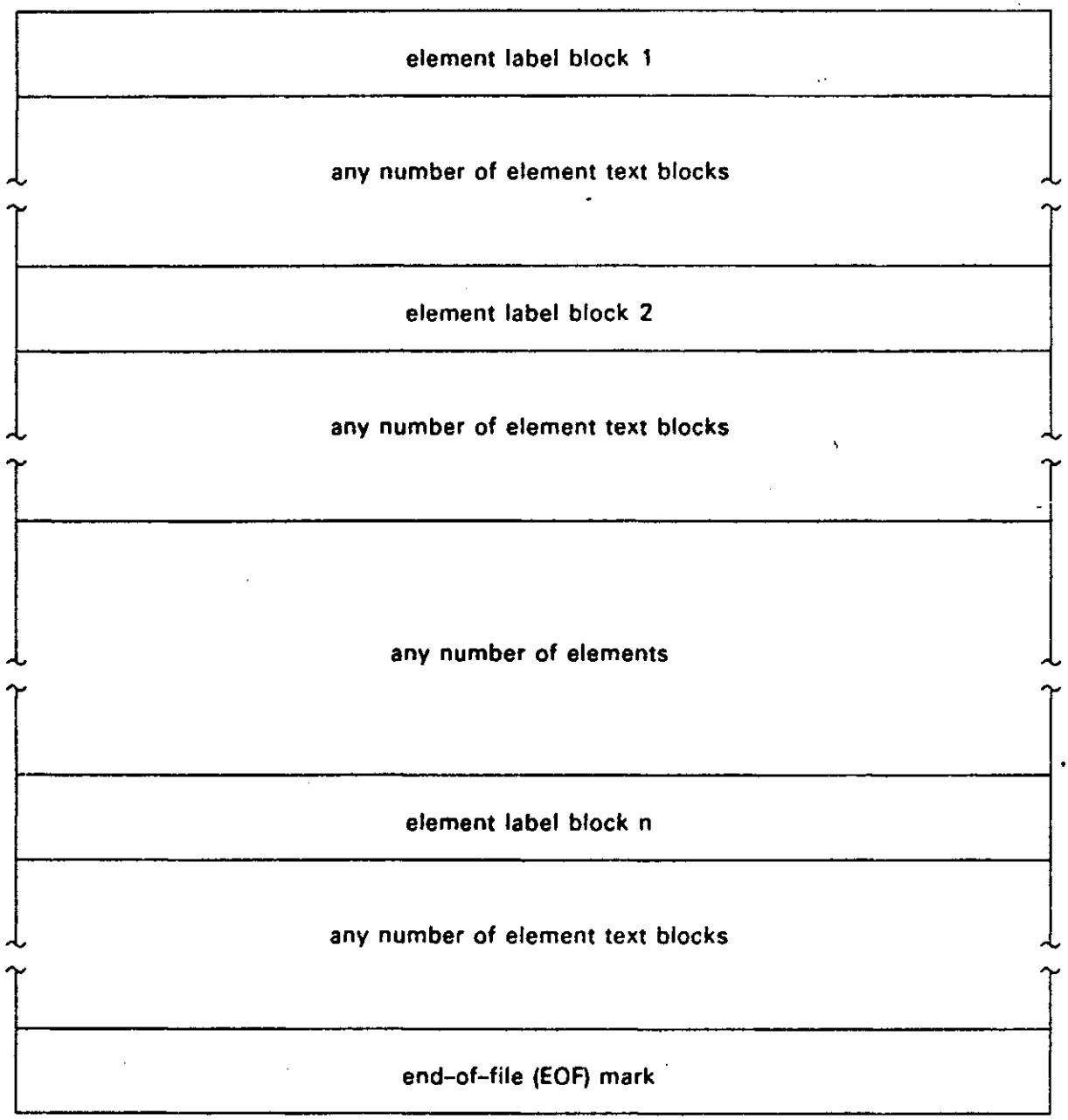


Figure 3-3. FURPUR Element File Format

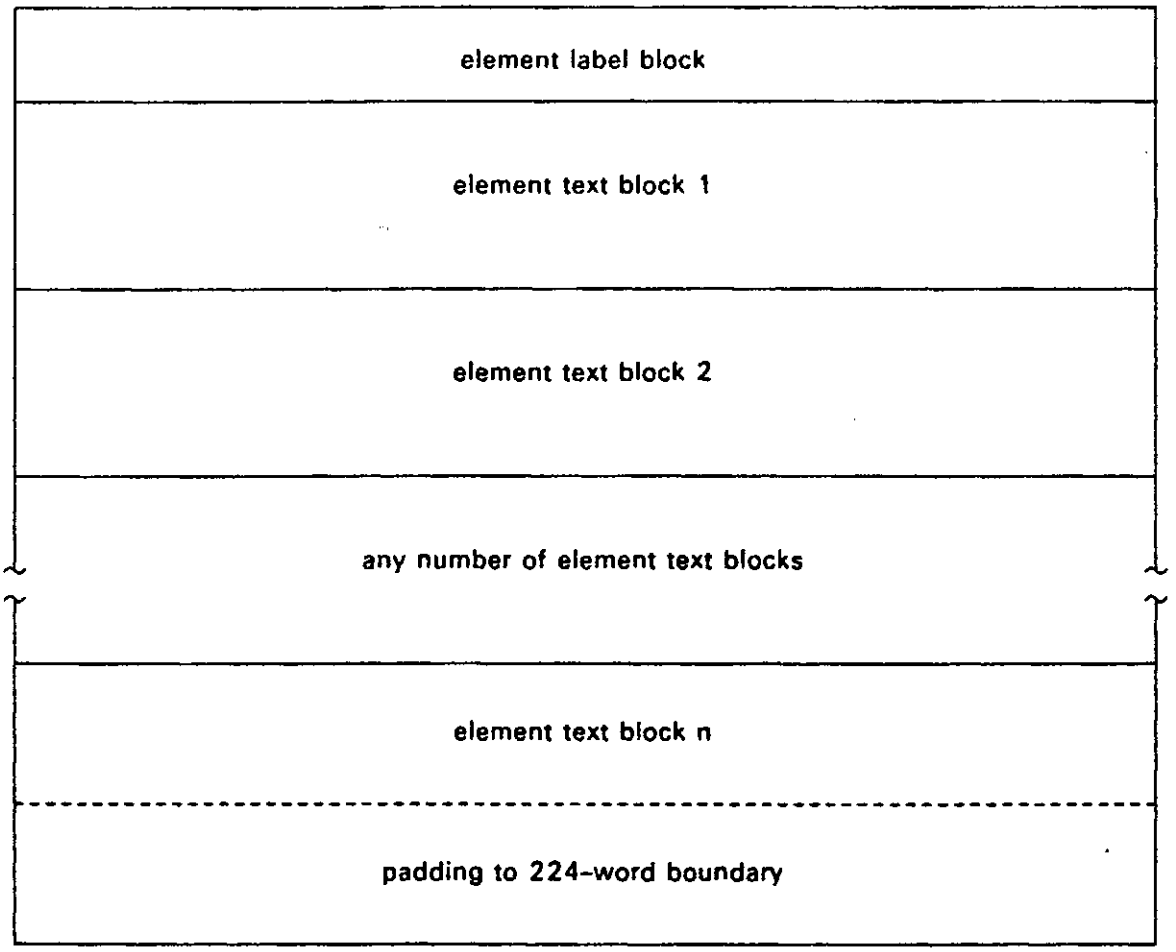


Figure 3-4. Element in Element File Format

3.4. Multireel Files

The FURPUR processor (see FURPUR Programmer Reference, UP-8724 (see Preface)) automatically generates and checks for end-of-reel sentinels.

The commands that write on tape (@COPOUT and @COPY) generate an end-of-reel sentinel for unlabeled tapes when the hardware returns an end-of-tape status. The end-of-reel sentinel for unlabeled tapes has the following form:

Word 0	0541600000000	
Words 1-7	0044105117122	(ASCII "$EOR")
Words 8-16	0000000000000	

FURPUR uses this format for both 8–bit and 6–bit labeled tapes. When writing to a quarter–word MSA tape, FURPUR uses the equivalent of the 8–bit format read as quarter–word.

The commands that read tape (@COPIN, @COPY, @FIND, and @MOVE) check the block following each end–of–file (EOF) mark for an end–of–reel sentinel. The check is made for both labeled and unlabeled tapes with one exception. A move across single–reel tapes does not read the next block after an EOF mark. Thus, files on a single–reel tape may have different file identifiers than the one used to assign the file.

The current end–of–reel sentinel check is for 054 in S1 of Word 0 in the first block after an EOF mark. If the sentinel is found processing continues on the next reel. If Word 0 does not contain an end–of–reel sentinel, FURPUR positions the tape between the EOF mark and the first block or, for a move (@MOVE) with a remaining file count above 0, continues on to the end of the next file.

NOTES:

1. *Limiting the end–of–reel check to 054 in S1 of Word 0 keeps FURPUR level 28R1 compatible with tapes written by earlier levels of FURPUR. Subsequent levels of FURPUR will expect the end–of–reel sentinel for unlabeled tapes to be as written by FURPUR level 28R1. The future FURPUR levels will not expect an end–of–reel sentinel on a labeled tape (instead, relying on the tape labeling information to determine when a tape swap is necessary).*

2. *Future FURPUR levels will not read unlabeled multireel tapes produced by the symbiont complex since they still use the old format of end–of–tape.*

3. *Not all levels of FURPUR handle multireel Processor Common Input/Output System (PCIOS) tape files because different levels use different methods of indicating end–of–tape..*

4. System Data Format (SDF)

System Data Format (SDF) is a sequential access, variable record length file format that is used by Series 1100 System software for some types of data storage. It is primarily used to store symbolic data in symbolic elements or in data files. Symbolic elements of program files are assumed to use SDF. Some processors use SDF data files to store non-character (binary) data that is logically formatted into separate records.

SDF files on mass storage are a continuous set of sequential data. SDF files or elements can be blocked; if they are, the blocking is normally 224 words per block with a continuation control word (051) being used to split the record that crosses the block, or a bypass control record (040) used to pad the unused part of the block. Some tape files (symbiont tapes) can be blocked at 1792 words per physical tape block. In addition, each tape block can be internally blocked in 224-word blocks.

Data is recorded in variable-length records consisting of a control word followed by zero or more data words. There are two different types of SDF records:

1. Data Records

   A data record is any record whose control word does not have bit 35 set. The remaining portion of T1 of the control word contains the length of the record. A maximum record length of 2047 words may be defined with one control word. The contents of the remaining portion of the control word vary depending on the specific type of SDF file or element.

2. Control Records

   Control records provide special control information as needed by the system components processing the file or element. A control record has bit 35 set in the control word. The control record contains a code in the range 040-077 in S1 of the control word. If an image follows the control word, its length is contained in S2. If no image follows, the content of S2 is zero. The maximum length of a control record that may be defined with one control word is 63 words.

4.1. SDF Types

SDF files and elements have types associated with them that determine the interpretation of parts of the control words. The SDF type (see Table 4-1) is identified by a Fieldata value in S3 of the control word for the first record of the file, which is always a label control record (050).

Table 4-1. SDF Types

Origin of SDF File or Element	S3 of 050 Control Record	Explanation
Unspecified	@	Unspecified file or element type. See 4.1.1.
Input Symbiont	C	Produced by the input symbiont complex via READ$. Distinguished from other "C" types (Output Symbiont) by usage and label-record length. See 4.1.2.1.
Output Symbiont	P or C	Produced by the cooperative symbiont complex via ERs SYMB$, PRINT$, PUNCH$, etc. (P=print file; C=punch file.) See 4.1.2.2.
@FILE (Symbiont)	I	Produced by a @FILE control statement. See 4.1.2.1.
FORTRAN V Library	F	FORTRAN V library data file. See 4.1.3.
General Symbolic	S	General symbolic file or element. See 4.1.4.
PCIOS	X	Processor Common I/O System (PCIOS) created file or element. See 4.1.5.

4.1.1. Unspecified Files (Type @)

CTS creates untyped SDF files and elements. Before definition of the general symbolic (S) type, general symbolic files and elements were also untyped. Now an unspecified type indicates that only the image length field of the data record control word contains valid information. Any information contained in bits 23–0 of data record control words has meaning only for the application that created the SDF file or element.

4.1.2. Symbiont Complex Files (Types C, I, and P)

The symbiont complex is a means of spooling input and output data streams to mass storage and, as such, is closely associated with mass storage files. There are three parts to the symbiont complex: input symbionts, cooperative symbionts, and output symbionts.

- input symbionts — create READ$ files for the cooperative symbionts to read the runstream from, and @FILE files for processors and runstreams to use.
- cooperative symbionts — read READ$ and other SDF (@ADD, READA$) files and produce PRINT$ and PUNCH$ files.
- output symbionts — read PRINT$ or PUNCH$ files and produce output paper.

All of the files used or produced by the symbiont complex are SDF files. These files share the common SDF record characteristics for differentiating data and control records, location of length fields, and definition of certain control records. Device or user-specified control records are, in general, unique control record types so that other users of the same file can bypass unknown information.

4.1.2.1. READ$ and @FILE Produced Files

The input symbiont complex produces two types of files: the READ$ file (which contains a runstream to be executed) and the @FILE file (which is a means of producing a file for later use by processors or runstreams). These two file types are identical except for the label block. The name of the READ$ file is always SYS$*READ$X*run-id*, while the @FILE file has a user-specified name. Except in the section on label control records (050), these two file types are discussed together in this manual.

4.1.2.2. PRINT$ and PUNCH$ Files

The cooperative symbiont complex produces print and punch output files via Executive Requests PRINT$, PUNCH$, and SYMB$ (and variations of these). These files are then placed on system queues for printing or punching. These files are identified by the label control word, which is different from other SDF and symbiont files as follows:

- Standard print files only contain data records and 042, 050, 051, 054, 060, 061, and 077 records. T2 and T3 of the data control word have meanings that are unique to print files.
- Standard punch files only contain data records and 042, 050, 051, 054, 070, and 077 control records. T3 of the data control word has meanings that are unique to punch files.

4.1.2.3. Symbiont Use of Other SDF File Types

The cooperative symbiont complex can use any type of SDF file or element and is sensitive to the cycle information that is available in the general symbolic type. If the data is not character data, the symbiont complex assumes character data unless a punch control record (070) is encountered that indicates a shift to binary data.

The punch output symbionts can use any SDF type but limit the amount of recovery that occurs, the accuracy of keyin responses, and may produce wrong results following repunches when punching mixed mode files.

The print output symbionts can print the data records from any SDF file, but the printout is limited if the file is not a symbiont print file. The print symbiont assumes a single space prior to every data record for non-print files. File error recovery is inhibited. Symbiont keyins produce estimated values instead of accurate values for some responses. Translate table usage is not available. Reprints may produce the wrong results in mixed mode (ASCII/Fieldata) files.

4.1.3. FORTRAN V Library Files (Type F)

The FORTRAN V common bank I/O library uses SDF files to control formatted and nonformatted records for mass storage and tape. This function has been replaced by the PCIOS interface for ASCII FORTRAN. See 4.1.5.

The FORTRAN V SDF files may differ depending on the release level. Information on the system data formats for FORTRAN V can be found in FORTRAN V Library, Programmer Reference, UP-7876, and FORTRAN V Common Bank I/O Library Programmer Reference, UP-8204 (see Preface).

4.1.4. General Symbolic Files (Type S)

The general symbolic file type is created by processors that use the source input/output routine (SIR$). This type of SDF file is created by Series 1100 Operating System software such as the language processors, ED, DATA, ELT, and PDP processors, etc. The S type indicates that bits 23–0 of the data record control words contain element cycle information. For SDF type S files, this information is not used. Control record types 042, 050, 052, and 077 are contained in SDF type S files.

4.1.5. PCIOS Files (Type X)

The Series 1100 Processor Common Input/Output System (PCIOS) is used with ASCII COBOL, ASCII FORTRAN, PL/I, QLP 1100, RPG 1100, and SORT/MERGE processors to produce compatible data files.

The SDF files produced by PCIOS contain:

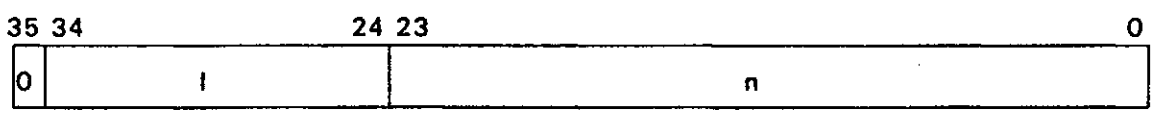
- An SDF label control record (050). If it is a direct file, sector zero contains the label. For sequential files, the label record is the first record of the first data block of each file; more than one SDF file may be stacked in a mass storage file.
- An SDF end-of-file (EOF) record (077), which is the last record of the file. The EOF record is the last record of the last block.
- SDF end-of-reel (EOR) records (054) when the file is a multi-reel file. The EOR record is the last record of the last block of every reel but the last.
- SDF bypass records (040). If the file is sequential, bypass records are used in the last block to force the EOF or EOR record into the last word of the last block. If the file is direct, sectors 1, 2, and 3 contain bypass records, and bypass records are also used in the last block to force the EOF record into the last word of the last block.
- SDF data records which may or may not be segmented. If the file is direct, the records are segmented at 2047 words (see 4.2.5). If the file is sequential, the records are segmented at the size specified by the Processor Interface Module (PIM) in the FCSS field of the File Control Table (FCT).

A record control word precedes each record or record segment.

A deleted or dummy record is identified by a 077 code in S5 of the record control word.

4.2. Data Record Formats

The general control word format for data records is:



where:

bit 35 Is always zero indicating that this is a data record.

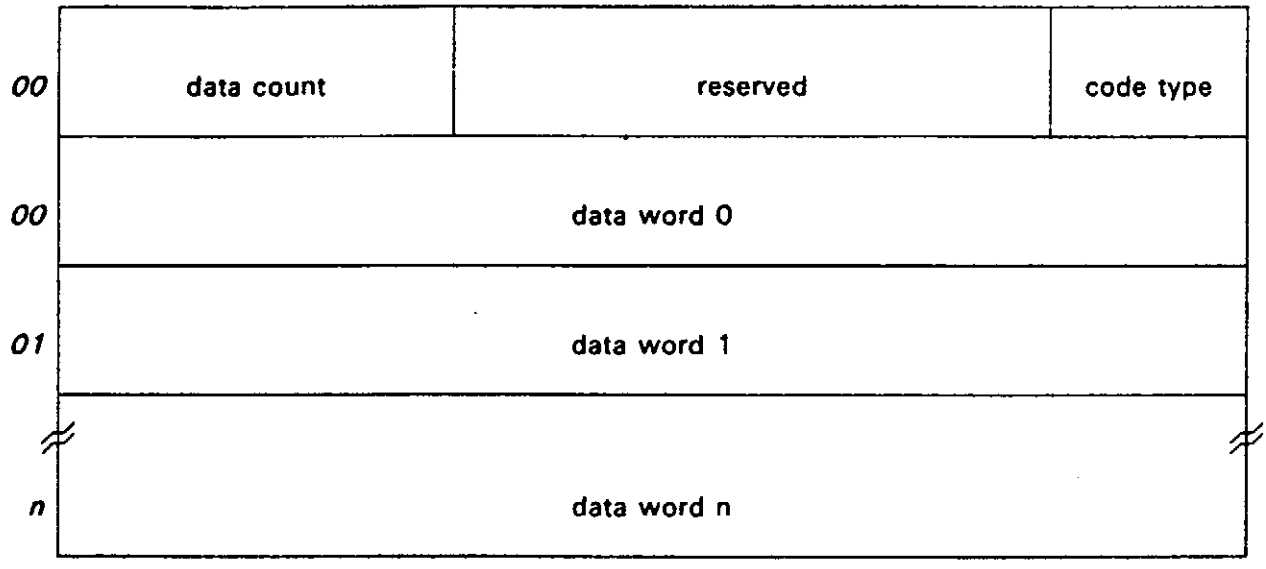
l — The length of the first segment of this data record. The data record can be continued by use of continuation control records (051). A maximum length for the first data record part is 2047 (03777), or the block size minus one, whichever is smaller.

n — The information in bits 0 to 23 of the data control word are variable depending on the code type as determined from the initial label control record. (See 4.3.4.)

4.2.1. READ$ and @FILE Data Records

The format for READ$ and @FILE data records is shown in Table 4-2.

Table 4-2. READ$ and @FILE Data Record Format



where:

data count — Is the number of words of data that follow. The allowable values are:

0-223

This is limited to 223 words because all symbiont files are blocked with 224-word blocks. If a longer record is required, or if a record spans blocks, then continuation control records (051) are used to complete the record.

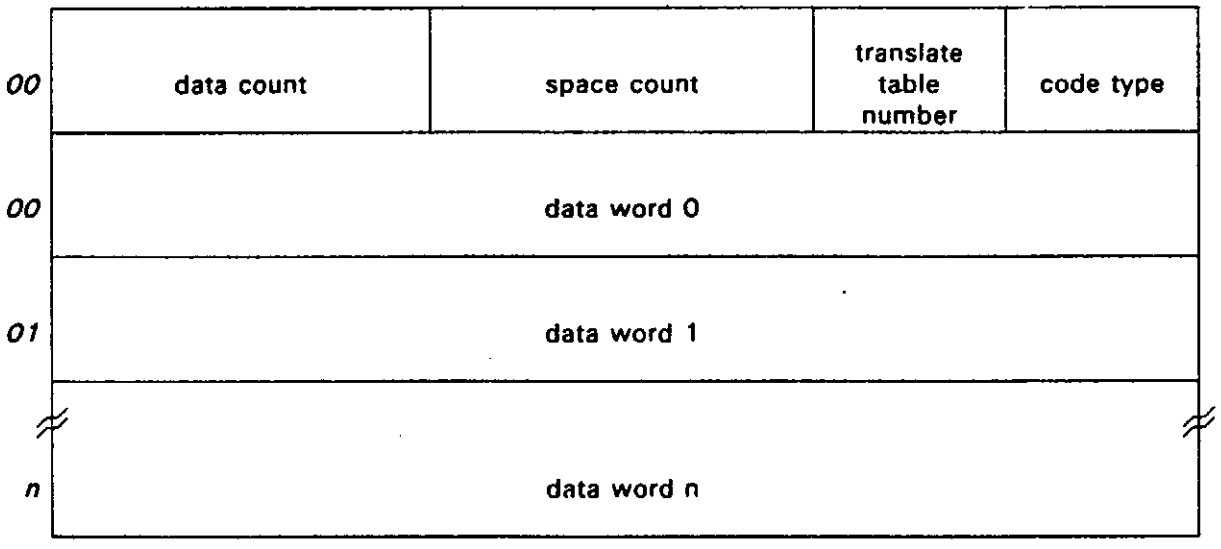
code type — This indicates the format (9 bits/character or 6 bits/character) and encoding of the characters (ASCII, Fieldata, etc.) in the data portion of the record. The data portion of these records is assumed to be binary if the proper variation of the punch control record (070) precedes the data records. If the data is binary, the code type field is ignored.

00	Fieldata 6-bit characters
01	ASCII/ISO 9-bit characters
02	ASCII/APL 9-bit characters
03	EBCDIC 9-bit characters
04	Binary
020	JIS 16 18-bit characters
077	Mixed mode (combinations of the above) (see 4.2.1)

4.2.2. PRINT$ Data Records

The format for PRINT$ data records is shown in Table 4–3.

Table 4–3. PRINT$ Data Record Format



where:

data count — Is the length of the following data area in words. The allowable values are:

0–223

This field is limited to 223 since print files are blocked at 224 words per block.

space count	The spacing value associated with this record:	
	0–2047	Indicates spacing to be done prior to printing this record.
	2048–4094	Reserved for future development. Produces a page eject.
	4095	Indicates a page eject is to occur prior to printing this record.
translate table number	The translate table number associated with this record.	
code type	See code type in 4.2.1.	

During normal printing, the handler spaces the number of lines given in the spacing count unless spacing occurs over a page boundary. If spacing occurs over a page boundary, the handler ejects a page and spaces to the first printable line on the next page (even without data length, i.e., data count equal to 0).

A spacing count of 4095 without data causes only a page eject, but if followed by a space-zero and print record, the space-zero record is printed on the first printable line of the next page as if it were a space-one and print record.

A spacing count of 4095 with data causes a page eject and the data is printed on the first printable line of the following page. This ensures compatibility with the way L, B, S, and M control records (alphanumeric print control record subtypes (060)) eject pages and allows the symbiont handlers to eject to the physical home paper position for these page ejects. A negative spacing with no data, followed by a space-*x*-and-print record ($x \neq$ zero), prints on line *x* on the following page.

If absolute line spacing has been specified, the spacing is handled differently. The spacing is still truncated to a configured value (PRSPMX - see Executive System Generation User Guide, UP-8448; see Preface) if it is greater than 2047, but any spacing value except 4095 causes exactly that amount of spacing independent of page boundaries and margins. A spacing of 4095 is handled as it is during a normal print operation.

4.2.3. PUNCH$ Data Records

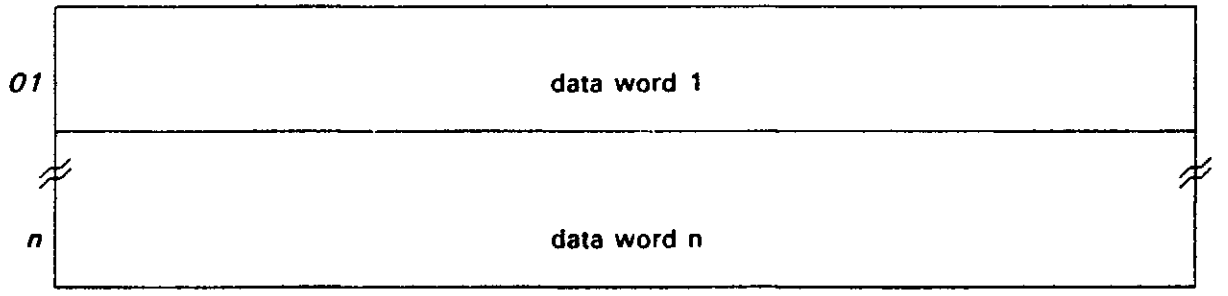
The format for PUNCH$ data records is shown in Table 4-4.

Table 4-4. PUNCH$ Data Record Format

00	data count	reserved	translate table number	code type
00	data word 0			

(continued)

Table 4–4. PUNCH$ Data Record Format (continued)



where:

data count	Is the number of words of data that follow. The allowable values are: 0–223 This is limited to 223 words because all symbiont files are blocked with 224-word blocks. If a longer record is required, or if a record spans blocks, then continuation control records (051) are used to complete the record.
translate table number	This is a value that may be used by some output devices to determine how to interpret the data in the record. It is normally used in conjunction with the code type field.
code type	This is a value that indicates the format (9 bits/character or 6 bits/character) and encoding of the characters (ASCII, Fieldata, etc.) in the data portion of the record. The data may also be assumed to be binary if the proper punch control record (070) precedes the data records. If the data is binary, then the translate table and code type fields are ignored. Punch control records (070) also affect the translation of the data when it is sent to the device. For example, the ASCII data in the record may be punched as Fieldata card images. (See code type in 4.2.1 for possible values.)

4.2.4. General Symbolic Data Records

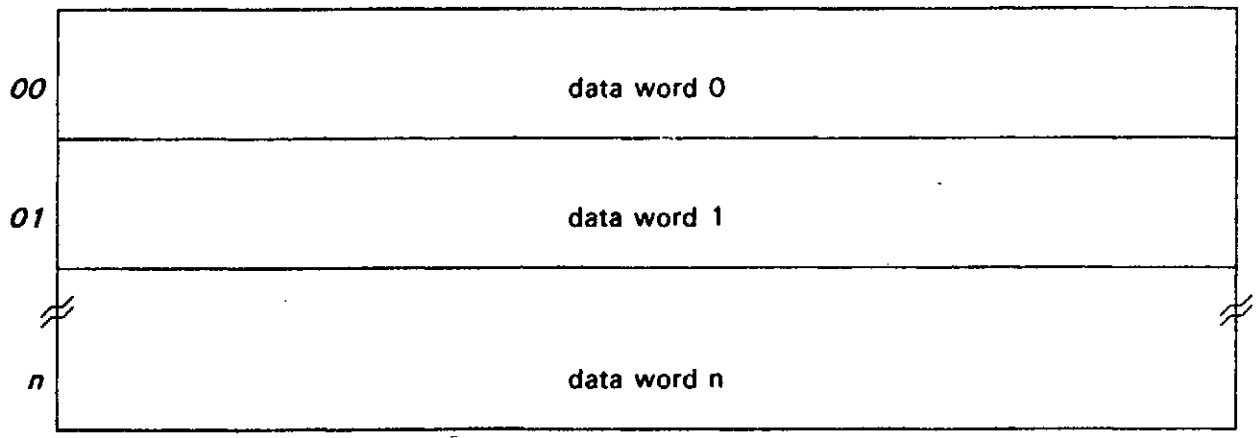
The format for general symbolic data records is shown in Table 4–5.

Table 4–5. General Symbolic Data Record Format

00	data count	delete flag	deleted cycle	add flag	add cycle

(continued)

Table 4-5. General Symbolic Data Record Format (continued)



where:

data count	Is the number of words of data that follow. The allowable values are: 0–2047 This is limited to 2047 words because bit 35 is used to indicate a control record. If a longer record is required, continuation records (051) are used to complete the record.
delete flag	A flag indicating that records were deleted before this record on the last update.
delete cycle	The cycle at which this record was deleted.
add flag	A flag indicating that this record was added on this update.
add cycle	The cycle at which this record was added.

4.2.5. PCIOS Data Records

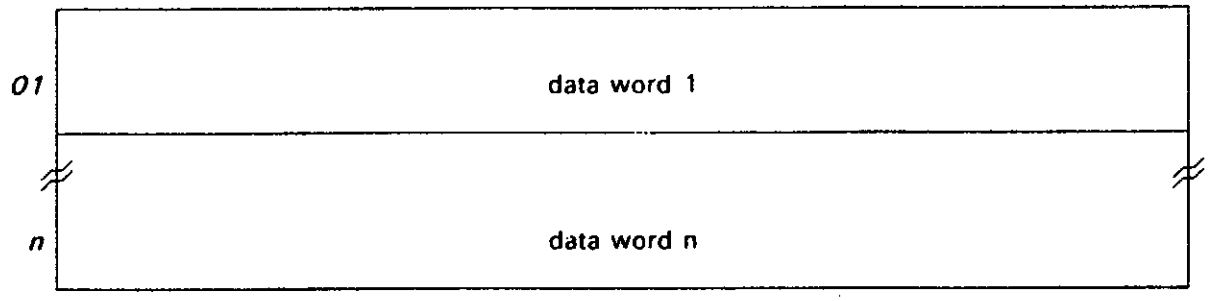
The format for PCIOS data records is shown in Table 4–6.

Table 4–6. PCIOS Data Record Format

00	data count	previous length	PCIOS caller dependent	segmentation flag
00	data word 0			

(continued)

Table 4–6. PCIOS Data Record Format (continued)



where:

data count — Is the number of words of data that follow. The allowable values are:

0–2047

This is limited to 2047 words because bit 35 is used to indicate a control record. If a longer record is required, the segmentation flag is used to segment the record data.

previous length — The length of the previous record.

PCIOS caller dependent — This cell is 63 for deleted or dummy records and is a variable value from 0 to 62 (depending on the originator) for other records.

If the originator is 1 (ASCII FORTRAN level 8R1 and lower), this is the FORTRAN record format:

01 Formatted record
02 Unformatted record
03 Namelist record
04 List–directed record
077 Dummy record for direct access.

If the file originator is 2 (PL/I), 3 (ASCII FORTRAN level 9R1 or higher) 1 or 4 (ASCII COBOL), this is the number of bits in the last word of the record.

segmentation flag — This indicates how this record is segmented:

0 This is the only segment for this record.

1 This is the first segment of this record.

2 This is the last segment of this record.

3 This is an intermediate segment of this record.

PCIOS data records may not always contain symbolic data. The means of determining the type of data when it is not symbolic varies depending on the file originator.

PCIOS does not normally produce continuation control records, but does accept them in input files. Segmentation of large records is handled by using multiple data records with the segmentation flag set appropriately. This means that some processors (such as the symbionts) may not handle large PCIOS records properly.

4.3. Control Record Formats

4.3.1. General

A control record contains a code in the range 040-077 in S1 of the control word. The currently defined control codes (octal values) are:

040 Bypass record:

This indicates that this image is to be skipped. S2 contains the number of words to skip. Skip to the next control word. See 4.3.2 for format.

042 Code type change record:

This is used when an SDF file or element contains images in more than one character code (e.g., ASCII, Fieldata, JIS); indicates a switch to the code type defined by S6. S2 is usually 0. If the file is a symbiont file, this indicates a change in either S5 (translate table number), or S6 (code type), or both. Format is shown in 4.3.3.

043 FORTRAN V backspace record:

S3 contains the length of the previous image. See 4.1.3 for additional information.

050 Label control record:

This is the initial control word of an SDF file or element. If a label follows, its length is in S2. The image format is dependent on the creating routine. S3 specifies the SDF type of file or element (see Table 4-1). S6 specifies the code type of the following images in the file or element as does the 042 control image. Formats are shown in 4.3.4.

051 Continuation record:

This indicates the following record is a continuation of the previous record. S2 specifies the number of words in this section of the record. This is used in symbiont files at the start of a 224-word block when an image spans blocks and to allow large control or data images. The format is shown in 4.3.5.

052 Change record (SIR$):

If SIR$ (see UP-8728) applies corrections to the input, the output will contain 052 control records. If H2 is nonzero, then H2 equals the number of images deleted before the next image in the last update. If H2 equals 0, this is a partial-line-change or line-change statement whose code type is the same as the current SDF input (described in SYSLIB Programmer Reference, UP-8730, see Preface). The format for 052 records is shown in 4.3.6.

053 CTS/HVTS line number record:

See 4.3.7 for format.

054 End-of-reel record:

This is used for multireel tape files to indicate a tape swap is required at the end of reel. See 4.3.8 for format.

060 Print control record:

This is used in symbiont print files. It contains alphanumeric character print control information (free format character data). Subsection 4.3.9 shows these special formats.

061 Special Print control record:

This is used in print files. It is similar to 060 records, but contains binary non-character images. See 4.3.10. for formats.

070 Punch control record:

This is used in symbiont punch files. S2 contains the number of words (image) which compose the image submitted to the ERs SYMB$/PCHCN$ or their ASCII and alternate equivalents. This image is then interpreted when the file is output to determine any mode change required. For example, an ER PCHCN$ to change the character set to EBCDIC would produce the image 700100000000_8 105671606060_8. See 4.3.11 for format.

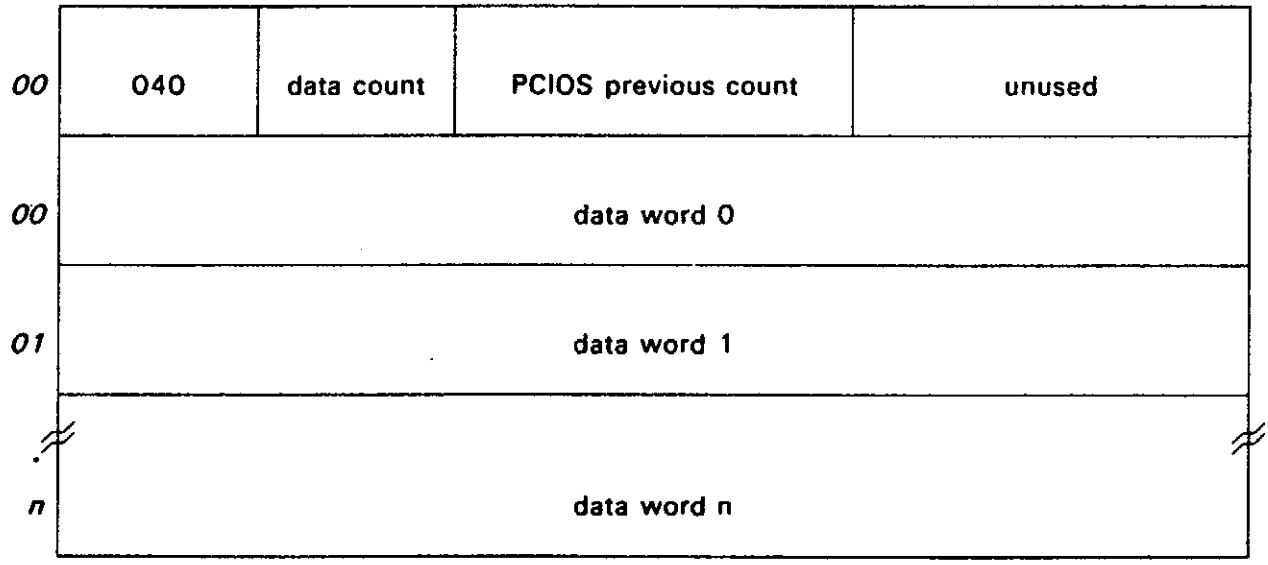
077 End-of-file record:

The end-of-file record indicates termination of the SDF file or element. See 4.3.12 for format.

4.3.2. Bypass Control Record (040)

The bypass control record format (control code 040) is shown in Table 4-7.

Table 4-7. Bypass Control Record Format (040)



where:

data count	This is the number of data words in this record. Although none of the data words are meaningful, the allowable values are: 0-63
PCIOS previous count	This is the length of the previous record. Only valid in PCIOS files or elements.

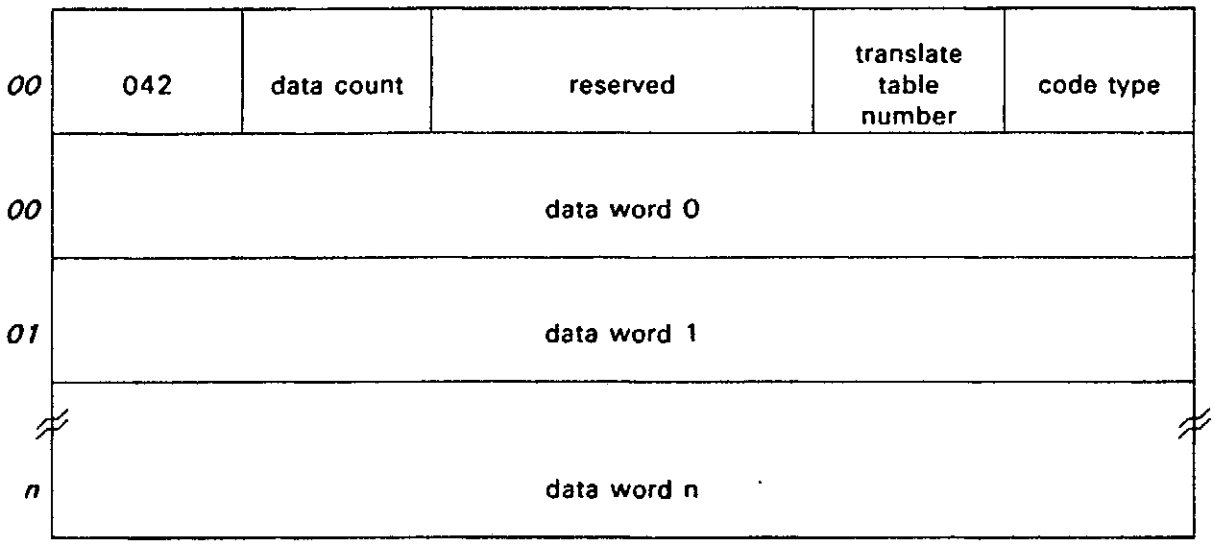
The bypass control record is used as a space filler in files. A symbiont uses this to pad the first block on a tape when doing a SV Keyin to save a nonstandard print file in order that a label can easily be added to the file. PCIOS uses this record type for padding the unused parts of sectors in direct access files, and to allow placement of the end-of-file and end-of-reel records at the desired positions in the file.

Although the data section is ignored, use of a data section allows for skipping large sections of a file with fewer bypass control records.

4.3.3. Code Type Change Control Record (042)

The code type change control record format (control code 042) is shown in Table 4–8.

Table 4–8. Code Type Change Control Record Format (042)



data count	This is the number of words, if any, in the data part of this record. Although the data words are not meaningful or interpreted if included, the allowable values are: 0–63
translate table number	This is used in standard print or punch files only (symbiont output) to indicate which translate table to use with the following data records. Undefined for input symbiont and non–symbiont code types.
code type	See code type in 4.2.1.

Code type change control records are used whenever a change is made to the code type (from S6 of data record control word) or translate table number (standard print/punch files only) of the data in the following records. Normally these records have a data count of 0 and no data words to skip.

These records are necessary for changing code type when accessing a file normally, but the use of these records does not imply that the code type indicator in S5 and S6 of data records is not needed in standard print files. The reprint/repunch logic requires that S5 and S6 of all records contain the code type in order to use the correct translation after moving backwards in the file.

Some processors may not respond to these images and therefore do not properly handle files with more than one code type.

4.3.4. Label Control Records (050)

Label control records are used as the initial record in a file to identify:

1. The file as SDF
2. The SDF type (see Table 4–1)
3. The code type (see 4.2.1.)

CTS, prior to level 7R1, used 050 records to store the line number. The 053 control record is now used for this purpose. The S1, S2, S3, and S6 fields of the record control word always contain valid information.

The code type specified in S6 of the control word of the label control record does not necessarily indicate the code type for any of the data records in the file. For example, a print file always starts with a label control record that specifies Fieldata but may be immediately followed by a change control record (042) which changes all following data records to ASCII.

4.3.4.1. READ$ Label Control Record (050)

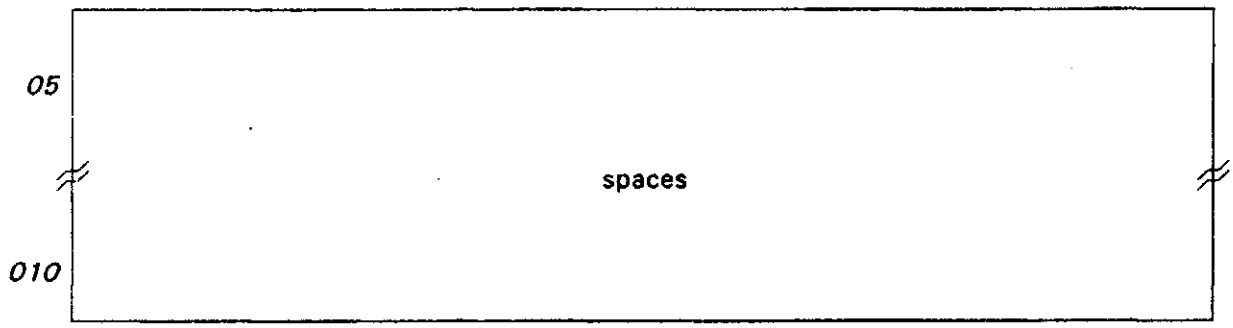
The READ$ label control record format is shown in Table 4–9.

Table 4–9. READ$ Label Control Record Format (050)

00	050	011	"C" SDF type	unused	code type
00	filename (READ$X*run-id*)				
01					
02	input device				
03	run–id				
04	time in TDATE$ format				

(continued)

Table 4-9. READ$ Label Control Record Format (050) (continued)



where:

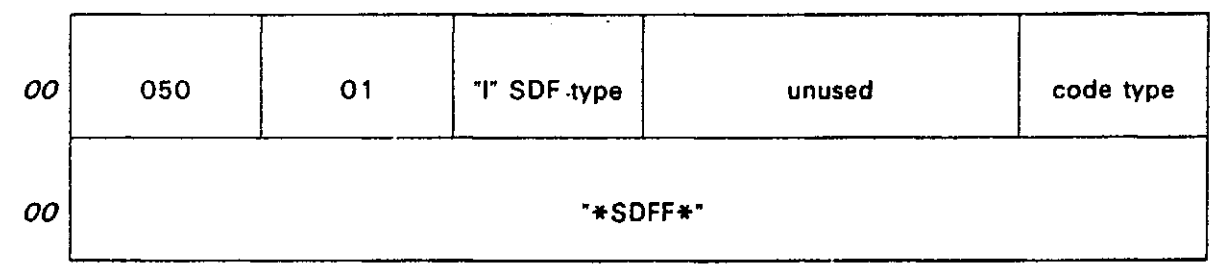
"C" SDF type	This is a Fieldata C that indicates this is a symbiont input file.
code type	See code type in 4.2.1.
filename	This is the filename of the file left-justified space-filled (LJSF) Fieldata (always READ$X*run-id*).
input device	This is the device name (left-justified space-filled Fieldata) of the device the file images were read on.
run-id	This is the run-id (left-justified space-filled Fieldata) from the @RUN control statement.
time	This is the time the images were read in TDATE$ format.
spaces	This is a word of Fieldata spaces.

This type of file is only created by the input symbiont complex. A file that also has an SDF type of "C" but that has a 20-word data area for the label record is produced by the cooperative symbiont complex as a punch file (see 4.3.4.3).

4.3.4.2. @FILE Label Control Record (050)

The @FILE label control record format is shown in Table 4-10.

Table 4-10. @FILE Label Control Record Format (050)



where:

code type — See code type in 4.2.1.

This type of file is only produced by the input symbiont complex when an @FILE control record is being processed.

4.3.4.3. PRINT$/PUNCH$ Label Control Records (050)

The label control record for standard print files has changed several times. There are two formats of print label control records that are recognized as standard by the current symbiont handlers (see Tables 4–11 and 4–12). The handler checks H1 of the control record to determine which type of file is being printed. If the file is of the old format, then S5 (translate table number) of the 042 record control words and S5 of data control words are not used during output.

Table 4–11. EXEC Level 33 through 36 PRINT$/PUNCH$ Label Record Format (050)

00	050	(011) length	"P"/"C" SDF type	part number	0	code type
00	filename (LJSF Fieldata)					
01	filename (continued)					
02	input device or queued device					
03	run-id (LJSF Fieldata)					
04	time in TDATE$ format					
05	user-id (LJSF Fieldata) (if configured, or else space)					
06	user-id (continued)					

(continued)

Table 4-11. EXEC Level 33 through 36 PRINT$/PUNCH$ Label Record Format (050) (continued)

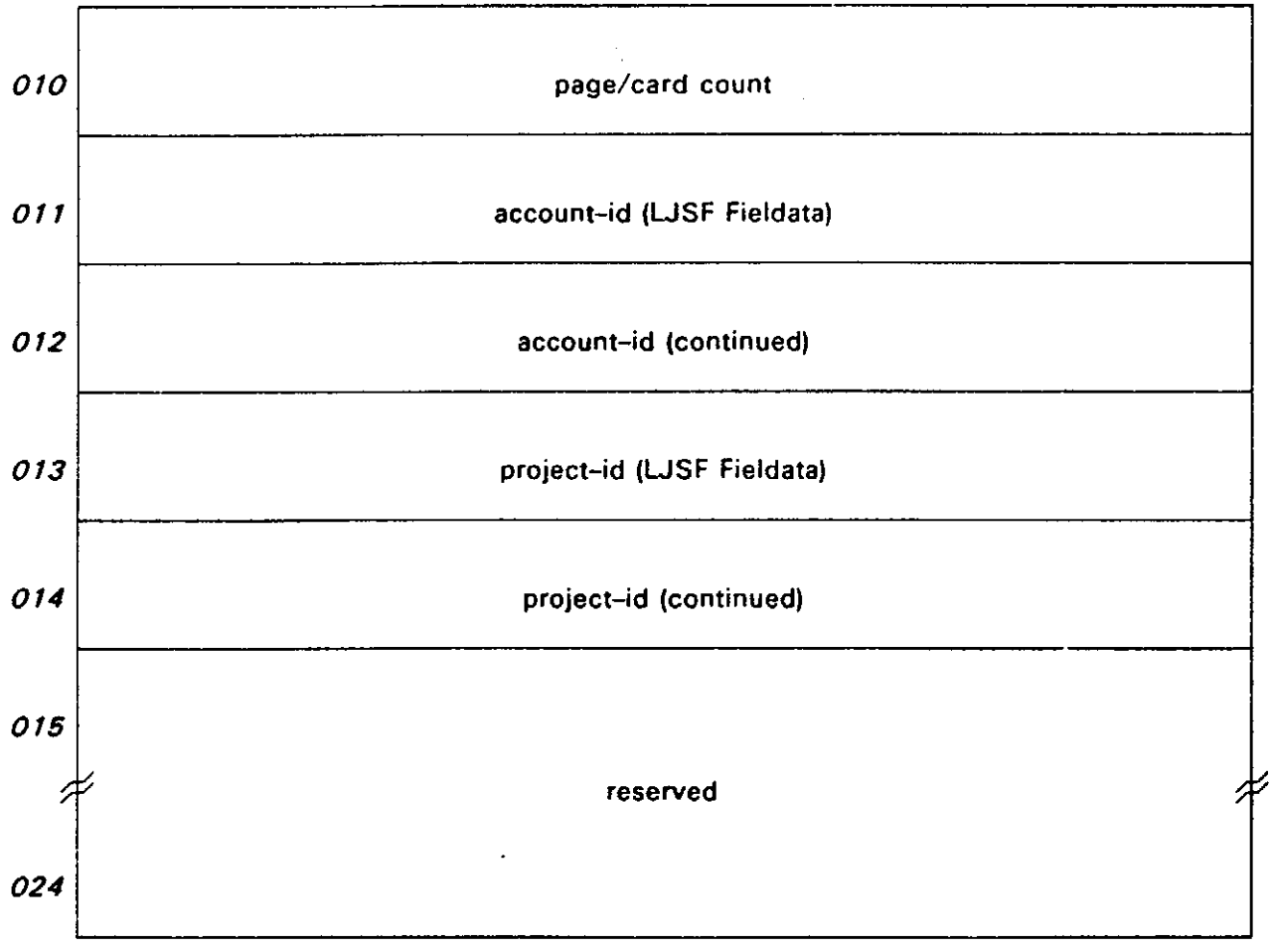
07	SV area
010	SV area (continued)

Table 4-12. 1100 OS Level 37 and higher PRINT$/PUNCH$ Label Record Format (050)

00	050	(024) length	"P"/"C" SDF type	part number	0	code type
00	filename (LJSF Fieldata)					
01	filename (continued)					
02	input device or queued device (LJSF Fieldata)					
03	run-id (LJSF Fieldata)					
04	time in TDATE$ format					
05	user-id (LJSF Fieldata) (if configured, or else space)					
06	user-id (continued)					
07	SV area					

(continued)

Table 4-12. 1100 OS Level 37 and higher PRINT$/PUNCH$ Label Record Format (050) (continued)



where:

length — This is the length in words of the data area of the record:

- 011 — The length of standard print file label blocks for EXEC levels 33 and 36.
- 024 — The length of standard print file label blocks for EXEC level 37 and higher.

"P"/"C" SDF type — A Fieldata "P" in S3 of the label control word indicates that this is a print file. A Fieldata "C" in S3 of the label control word indicates that this is a punch file.

part number — This is the breakpoint part number of this file. Used for PRINT$/PUNCH$ files (PR@*numrun-id* files only) that are breakpointed.

code type — See code type in 4.2.1.

filename — This is the filename in Fieldata (12 characters left-justified space-filled). This is the internal filename specified on the @BRKPT control statement or in the alternate ER packet for user files.

input device	This is the name of the input device that initiated the run that created the file. If the file was saved (SV keyin), then this is either the device name that the file was queued to when the SV keyin occurred or spaces (if the file was restored (SR keyin)). The name is in left-justified space-filled Fieldata.
run-id	This is the run-id of the run that created the file (6 characters left-justified space-filled Fieldata).
date	This is the date the file was created in TDATE$ format.
user-id	This is the user-id of the run that created the file (12 characters left-justified space-filled Fieldata). May be spaces depending on the system configuration.
SV area	This is an area that is used by the SV keyin to save information needed by SR keyin to restore the print file to the proper queue and priority.
page/card count	This is the number of pages or images that the operating system estimated were put into the file:

0	Implies that this was a tape file or the EXEC was unable to update the label block at the closing of the file.
1 to 2^{35}	This is the operating system's page/card estimate.
050505050505	This is a value used as a flag by the SV keyin to indicate that there is not a good size estimate.

account-id	This is the account number of the run that created the file (12 characters left-justified space-filled Fieldata).
project-id	This is the project-id of the run that created the file (12 characters left-justified space-filled Fieldata). This is not necessarily the file qualifier.
reserved	These words are reserved for future use by the EXEC and should be set to zero.

4.3.4.4. General Symbolic Label Control Record (050)

The general symbolic label control record format is shown in Table 4-13.

Table 4-13. General Symbolic Label Control Record Format (050)

00	050	01	"S" SDF type	unused	code type
00	"*SDFF*"				

where:

code type See code type in 4.2.1.

4.3.4.5. PCIOS Label Control Record (050)

The Processor Common I/O System (PCIOS) label control record format is shown in Table 4-14.

Table 4-14. PCIOS Label Control Record Format (050)

00	050	(033) length	"X" SDF type	unused	code type
01	filename (LJSF Fieldata)				
02					
03	type	originator	level	recovery offset	
04	block size			record size	
05	date				
06	address of next file				

(continued)

Table 4-14. PCIOS Label Control Record Format (050) (continued)

07	largest relative record key allowed			
010	largest relative record key written			
011	rel-rec-1-offset			segment size
012	FORTRAN format code	skeleton-ization flag	0	record length
013	address of previous file			
014	reserved (0)			
015	reserved (0)			
016	(words 016 through 033 are written as they appear in the label area at open time)			
033				

where:

code type — This is the same as code type in 4.2.1 except that the values can only be:

- 00 Fieldata 6-bit characters
- 01 ASCII/ISO 9-bit characters

filename — This is the internal filename in Fieldata (12 characters left-justified space-filled) when the file was created.

type — This is the type of PCIOS file:

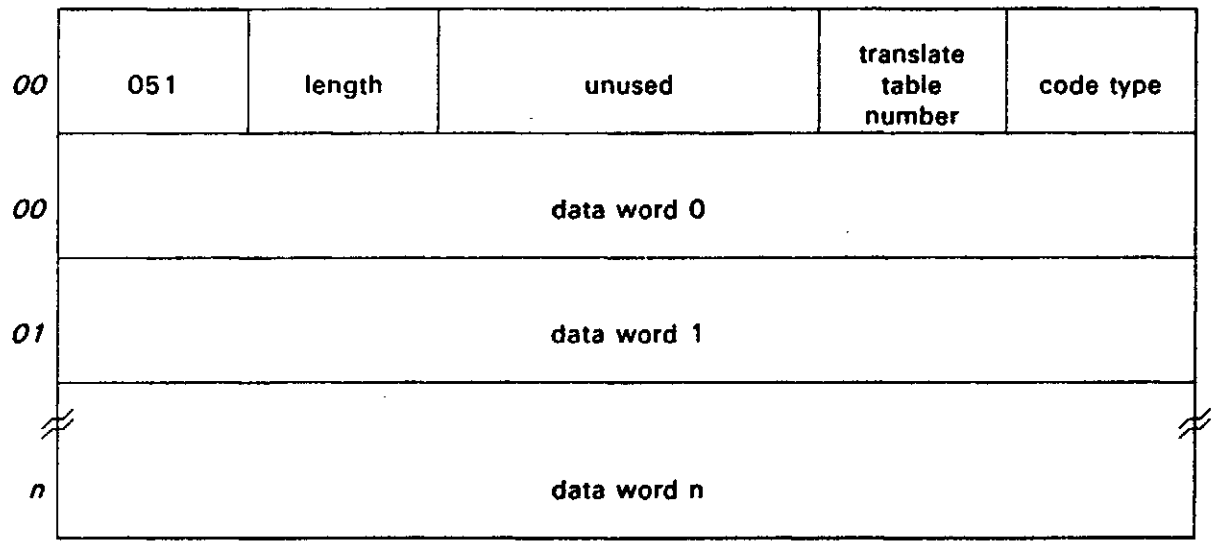
- 01 sequential
- 02 direct

originator	This indicates the file originator: 00 Unknown 01 FTN (Level 8R1 and earlier) 02 PL/I 03 FTN (9R1 and later) and APL 04 ACOB, RPG, and QLP 05 Mixture of 01, with 02, 03, or 04.
level	If set, this indicates that the file was created or extended by PCIOS level 4R1 or higher.
recovery offset	This is valid only if level is set. This is used for output extend recovery in the event of system failure during an extend operation.
block size	This specifies the block size in words of the file.
record size	This specifies the record size in words of the file.
date	This contains the date when the file was created.
address of next file	This contains the sector address of the next file for sequential stacked files on mass storage. This field is 0 for SDF direct files.
largest relative record key allowed	This is the maximum relative record number which is allowed for the direct SDF file. This word is zero for sequential files.
largest relative record key written	This is the largest relative record number which is written in the direct SDF file. This word is zero for sequential files.
rel-rec-1-offset	This is valid only if level (S3 of Word 03) is set. PCIOS levels 4R1 and higher use this for calculating record location from the beginning of a file.
segment size	This indicates the segment size (in words) used when creating this file.
FORTRAN format code	This is the FORTRAN record format control code.
skeletonization flag	This is the skeletonization flag set at file creation time. It is set from the corresponding field in the file control table.
record length	This is the record length in characters of the file.
address of previous file	This is the address of the previous file if files are stacked on mass storage. For the first file, this is -0 since there is no previous file.

4.3.5. Continuation Control Record (051)

The continuation control record format is shown in Table 4–15.

Table 4–15. Continuation Control Record Format (051)



where:

length	The length of the data area of the record.
translate table number	The translate table that should be used with this record. (Only valid for standard print/punch files.)
code type	The type of character set in the data area. (Only valid for standard print/punch files.) See code type in 4.2.1.

The continuation control records are used to continue any record when the record length field of its control word is too small or to allow error recovery in standard print files when the record spans 224-word blocks. More than one of these records may be used consecutively.

For standard print file error recovery a control word is required at the start of each 224-word block of the file. This is normally accomplished by splitting any records that cross a 224-word boundary with a continuation record.

PCIOS uses a different means of continuing data images. See 4.2.5.

4.3.6. Line – Change Control Record – SIR$ (052)

The SIR$ Line – Change control record format is shown in Table 4–16.

Table 4–16. SIR$ Line – Change Control Record Format (052)

00	052	data count	0	number of deleted images
00	data word 0			
01	data word 1			
	≈			≈
n	data word n			

4.3.7. CTS/HVTS Line Number Control Record (053)

The CTS/HVTS line number control record format is shown in Table 4–17.

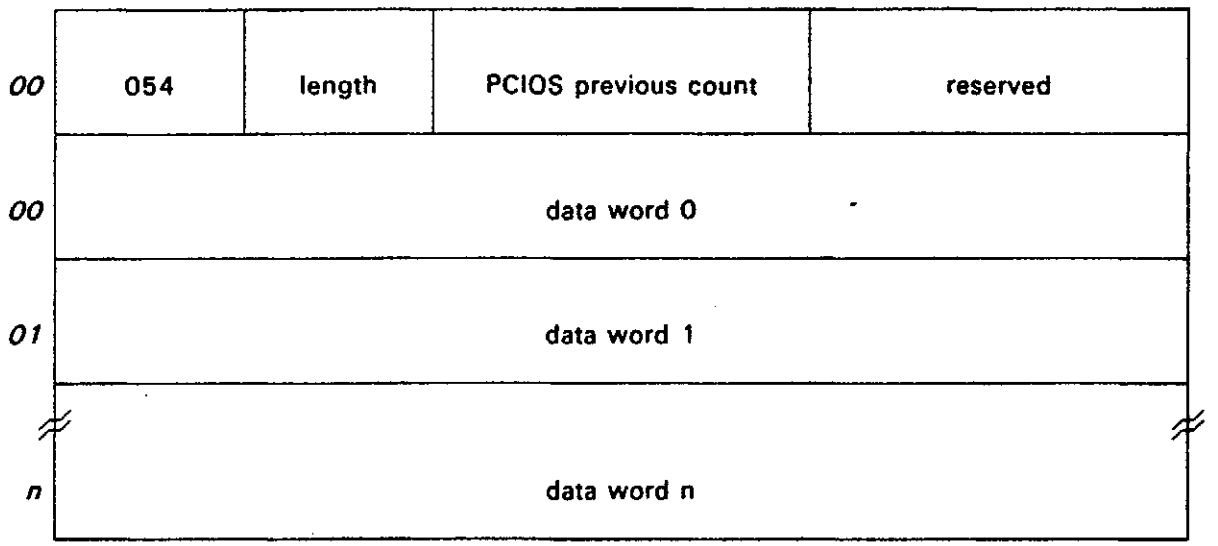
Table 4–17. CTS/HVTS Control Record Format (053)

00	053	0	0	line number

4.3.8. End-Of-Reel Control Record (054)

The end-of-reel control record format is shown in Table 4-18.

Table 4-18. End-of-Reel Control Record Format (054)



The end-of-reel (EOR) control record is inserted on an unlabeled tape after the end-of-tape (EOT) mark has been detected on a write.

The symbiont complex writes this record as the only record in a separate block that is written following a hardware end-of-file (EOF) mark. Therefore, processors reading unlabeled symbiont tapes must check the block following every EOF mark to verify the end of file. The EOR record on symbiont tapes is a 15-word record that contains 05416 in T1 of the first word. The remainder of the record is indeterminate.

FURPUR also places the EOR record in a separate tape block that is preceded by an EOF mark, but the data in the remainder of the record is unique. (See 3.4).

FURPUR and the symbiont complex do not use EOR records when the tape is labeled, but instead rely on the tape labeling EOR blocks.

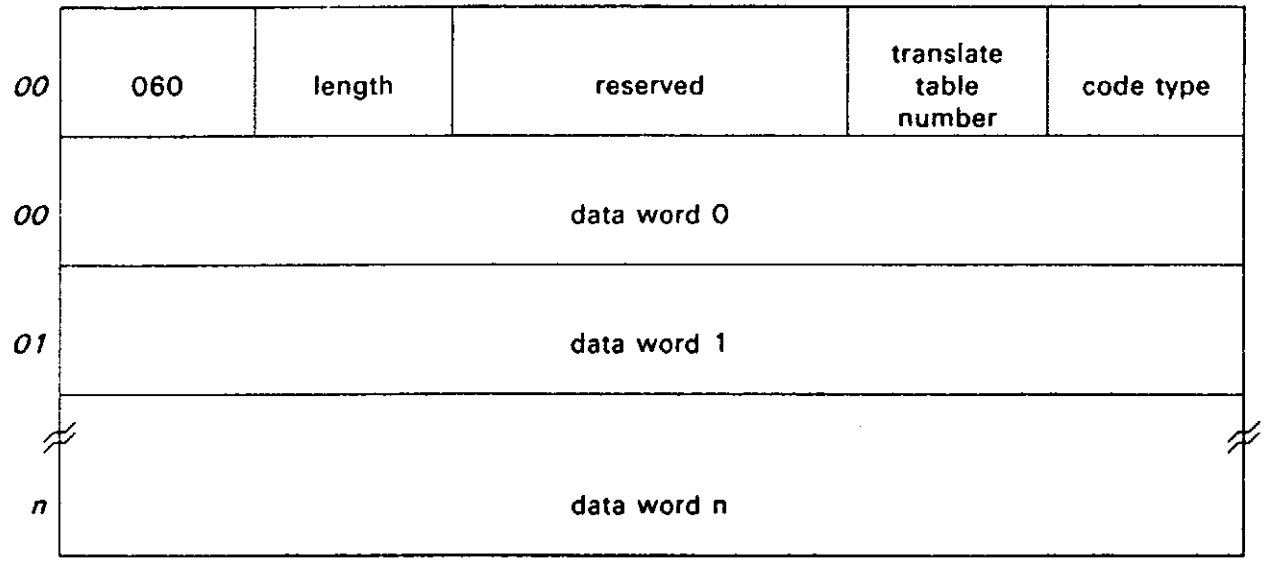
PCIOS places the EOR record in the main file as a normal file record instead of in a separate block. PCIOS may also place the EOR record in labeled tape files.

Because of the differing usage of the end-of-reel control record, PCIOS tape files are not compatible with FURPUR or the symbiont complex. In addition FURPUR and symbiont tape files are not compatible with PCIOS. In levels higher than 28R1 of FURPUR, symbiont unlabeled tape files are not compatible with FURPUR but FURPUR unlabeled tape files are still compatible with the symbionts.

4.3.9. ER PRTCN$ Control Records (060)

The ER PRTCN$ control record format is shown in Table 4-19.

Table 4-19. ER PRTCN$ Control Record Format (060)



The ER PRTCN$ control record consists of a control word with 060 in S1 and a special control record in the following several words. The special control word has the length of the special control record in S2 and the code type (1 for ASCII or 0 for Fieldata) in S6. The record consists of one or more print control functions to alter the printing of the file and has a maximum length of 63 words.

The ER PRTCN$ print control functions are interpreted separately and acted on as if they were in separate records so this manual treats them each individually. Separate statements in the same record are separated by periods. A space period space terminates the record scan. Therefore, the remainder of the record can be used as a comment field. Note that a period immediately following a nonspace character does not terminate the record even if followed by spaces.

These control records are bypassed when printing nonstandard print files.

See Executive System, Programmer Reference, UP-4144.26 (applicable version; see Preface) for a complete explanation of these functions.

4.3.9.1. LPI or Band (B) Control Functions

Format:

B[,*lpi*] Δ *ctid*[,*eltname*]

The LPI or band (B) control function record affects line spacing and print band usage. It is interpreted only for onsite printers and Nine Thousand Remote (NTR) printers.

If *lpi* is 6, 8, or *, a page eject equivalent to 4095 spacing without data occurs, followed by either:

- a request (via a simulated special forms request) to change the printer loop (1004 or 0768), or
- a change in the vertical format buffer (NTR, 0770, or 0776 printers).

The request or change is to the value specified if 6 or 8, or to the default device standard if an * was specified. If *lpi* is not specified, no change of density (LPI) occurs.

If *ctid* is specified, this is a request for a new load code or print band on an NTR, 0770, or 0776 printer; all other printers skip this part of this control record.

A request for a new print band causes an operator message to change the band and the transmission of the new load code to the printer, but does not result in any print spacing. Therefore, the operator should not home the paper while changing the band and the user could use several different bands to print one page.

The element name (*eltname*) on this control record is an optional field. If it is not included, the load code for the requested band is taken from the standard load code element. If *eltname* is specified, the load code is taken from SYS$*PRINTER$.*eltname*. The element format is shown in the EXEC Programmer Reference, UP-4144.2 (see Preface).

4.3.9.2. Logical Line (L) Control Function

Format:

L,*n*

The logical line (L) control function record is used to space to a specific printable line location on the page. It results in the same spacing for normal printing as it does when an M,X control function record has been executed. This control record positions the printer such that a "space one and print" record results in printing on the line specified after the comma. An L,*n* (*n* ≠ 0 or 1) control record at or beyond line *n* results in a page eject followed by the same spacing that would result from the L,*n* control record after a page eject. An L,*n* (*n* ≠ 0 or 1) control record when at line *m* (*m* < *n*) results in spacing where a space–one–and–print record prints on line *n* (spacing to printable line *n*–1 in general).

An L,0 or L,1 record results in spacing to the bottom printable line of the page if currently positioned above the bottom margin, in spacing to the bottom physical line on the page if inside the bottom margin, or no spacing if already at the home paper position. Note that since spacing to the bottom physical line on the page is not the same as a "home paper," a space–*n*–and–print after an L,0 or L,1 record results in printing on the first printable line of the next page, even if *n* is greater than 1, unless the printer was positioned at "home paper" before the L,0 or L,1 record.

4.3.9.3. Heading (H) Control Function

Format:

H,*options,page,text*

The heading (H) control function record allows the insertion of a heading record into the print file. This control record does not cause any special spacing to occur. It only inserts information into the file to be used the next time a top of page is encountered with enough top margin for a heading to be printed. The maximum size for a heading is 96 characters.

The *options* field can be N or X. An N results in the record text being ignored. An X suppresses the printing of the page and date information. A value in the *page* field is used as the starting page number if page numbering is specified. The page value is incremented even if page numbering is suppressed. Therefore, an H command that specifies page numbering but does not give a page value has a page number that starts numbering from the last H command that did specify a page value.

4.3.9.4. Special Forms (S) Control Function

Format:

S,*text*

The special forms (S) control function record is the equivalent of negative spacing without a record. It sends a message to the operator to place special forms in the printer. Even if the operator does not place new forms in the printer, the paper should be homed. A flag is set to cause a message at the end of the file to restore to standard forms.

4.3.9.5. Margin (M) Control Function

Format:

M,*length* –or– X,*top,bottom,lpi*

The margin (M) control function record is equivalent to negative spacing without a record, followed by a change of one or more of: page length, top margin, bottom margin, lines per inch, or method of computing spacing (X option). If page length is being changed, the *lpi* field must be coded or a B,*n* command must be entered along with this control record to keep the printouts which follow properly aligned.

An * may be used in place of any field and results in the device standard value being used. A blank field results in no change for that field. The *top* and *bottom* margin fields must be less than 63. The page *length* has a maximum determined by the printer used, or a configured value (PRSPMX), whichever is less (i.e., 192 lines on an 0770 printer). The *lpi* field must be a 6, 8, or *.

An M,X control record bypasses the normal end of page handling. It suppresses all headings, top margins, and bottom margins. The normal logic to truncate spacings that extend over page boundaries is inhibited, and the printer is spaced as far as requested, or a configured value (PRSPMX), whichever is less. A negative spacing is honored as a page eject, with the normal restrictions on the following record. All L,*n* commands are honored as if there are no top or bottom margins. This mode is terminated by doing another M command or B command. The *length* and *lpi* parameters used in this mode are those in effect at the time of the M,X control record.

The fields of the margin control function allow for up to 4 numeric decimal characters per field; if more are used, an error message results.

4.3.9.6. Error Recovery (I and A) Control Functions

Formats:

I

A

The error recovery (I and A) control function records have no effect on spacing; they only affect recovery from bad print records in the file. The I inhibits error recovery and the A allows error recovery. When error recovery is inhibited, any errors encountered that are file related result in termination of the print file with an error message.

4.3.9.7. Transparent (D) Control Function

Format:

D,@@*ctl*

This is a special RSI image and is ignored in print files. It is used when the PRINT$ file is a demand terminal.

4.3.9.8. Skip Break (R) Control Function

Format:

R

The skip break (R) control function record stops the reprint logic from skipping records when advancing through a print file (SM PRX Rnnn command). Spacing does not occur as a result of this record. Therefore, it should only be used when at the start of a page such as after a negative spacing, B, *lpi* or M,*length,top,bottom* commands, etc. When this record is encountered during a skip, a message is displayed at the system console.

4.3.9.9. Width (W) Control Function

Format:

W, *line-width*

The width (W) control function record allows for changing the maximum allowed width of printing. If the requested width is more than the maximum for the printer, the output records are truncated to the maximum for the printer in use. A W,* command returns to the device default for the printer in use. Spacing does not occur as a result of this record. The value *line-width* is the maximum number of characters divided by 6; it has a maximum value of 63 (378 characters).

4.3.9.10. User (U) Control Function

Reserved for site use.

4.3.10. Special Print Control Records - PRINT$ (061)

The PRINT$ special print control record (061) is inserted into a file via an EXEC function CN$ from the ER SYMB$ packet. Subsection 4.3.10.1 describes the first word of an 061 record which is common to all special print control records. Subsections 4.3.10.2 through 4.3.10.11 describe the subtypes that vary with each special print control function.

4.3.10.1. Common Parts of the Special Print Control Record

Table 4-20 illustrates the first two words of the 061 control record. The first word is common to all 061 records. Word 01 (the second word) varies with the subtype. These subtypes are described in 4.3.10.2 through 4.3.10.11.

Table 4-20. Special Print Control Record Common Sections (061)

00	061	length	0	0	translate table number	code type
00	subtype	varies				

where:

length — This is the total length in words of this part of this record. The allowable values are:

0–077

If the record spans a 224-word block or needs more than 077 words, it may be continued with one or more continuation control records (051).

translate table number — This is the translate table as specified by the ER SYMB$ that generated the record.

code type — This is the current code type as determined by the ER SYMB$ that generated the record.

subtype — This determines the function and format of the record. The allowable values are:

01–012

See the following subsections for individual descriptions.

The following subsections describe Words 01 through n of each subtype of the 061 control record.

- File or Element References

  Several 061 images allow specification of a filename or element name from which to retrieve the information. If the printer that is printing the file is being driven by a host system (such as a V77 System) that does not have partitioned data files (such as program files), then this field specifies the name of a system file that contains the data. If the printer that is printing the file is being driven by a host with partitioned data files (such as program files on a Series 1100 System), then an element name from a system standard file (normally SYS$*PRINTER$) is specified. On some systems it may be possible to modify the system file in which the elements are contained.

4.3.10.2. Load Translate Table (Subtype 01)

The load translate table special print control record format is shown in Table 4-21.

Table 4-21. Load Translate Table/Load Code Format (Subtype 01)

00	01	operation	translate table number	data length
01	data			
n	unused or data			

where:

operation — This determines the type of operation for this image.

- 0 — Implies this is a 0777 or similar printer translate table. This implies the first data format shown below.
- 1 — Implies this is a 0770 type load code buffer description. This implies the second data format below.
- 2 — Implies this is a 0776 type load code buffer description. This implies the second data format below.
- 3 — Implies this is a 0768 type load code buffer description. This implies the second data format below.

translate table number — This is the number of the translate table to be affected.

- 0–3 — The values allowable for the 0777 printer.
- 0–1 — The allowable values for 0768, 0770, or 0776 printers. 0 implies replace the Fieldata load code in use. 1 implies replace the ASCII load code in use.

data length — This is the number of quarter-word bytes of data. The allowable lengths for the data section are:

01–0402

The hardware requires at least one byte (the space byte) so the software also checks that it receives at least one byte to send. For

0768, 0770, or 0776 printers the data length should be exactly the number of characters on the print band plus two, or if dualing is desired, the number of characters on the band plus eleven.

data

This is the data for the translate table or load code buffer.

Format 1

An exact copy of a translate table as it would be sent to the printer in "A" format (9-bit bytes). The first byte is the space character value. The second byte is the underscore character value. This is followed by up to 256 bytes of data, each of which is the character generator location of the character that is to be printed when the number corresponding to this byte location is sent in the print data. Therefore, the third byte of data for a Fieldata translate table should be a pointer to the character generator location of the @ character. The ninth byte of data for a Fieldata translate table should be a pointer to the character generator location of the "A". Any locations that are not to be printed should contain an 0377 value.

Format 2

An exact copy of the load code buffer of a 0768, 0770, or 0776 printer in "A" format (9-bit bytes). The first byte is the cartridge-id. If different from that currently loaded, a console message changes bands and the other load code buffer (ASCII or Fieldata) is invalidated. If one of the load code buffers is invalidated, all data that is in that code is translated by the READ$ translate routines before printing.

The second byte is the space code if dualing is not active or the first byte of the dualing pairs if dualing is active. If dualing is active, there are four pairs of dualing bytes (see *1100 Series 0770 Printer Subsystem, Programmer Reference*, UP-8143, or *1100 Series 0776 Printer Subsystem, Programmer Reference*, UP-8485) and one data check dual, followed by the space character. Following the space character is an array of characters with a one-to-one correspondence to the characters on the print band.

The load translate table special print control record (01) is used to allow user-selectable loading of translate/load code tables for converting the user's internal codes into characters for printing. The translate table used (in standard print files) for the actual printing is that specified in S5 of the data record. The load code table used for printing is determined by the setting of bit 0 of S6 of the data record, the last 042 record, or the last 050 control record.

The offline system (0777) has a means of overriding the S5 test via a console keyin if print files created on older Series 1100 Operating Systems are to be printed. This always results in S6 specifying the translate table in addition to the character size.

The translate table used for nonstandard print files is determined by the file type field of the last 042 or 050 control record encountered in the file (table 0 for Fieldata and table 1 for ASCII).

The system default is a Fieldata translate table in translate table 0, an ASCII translate table in translate table 1, and no translate tables in translate tables 2 or 3.

There are several special uses for special translate tables in addition to different character sets (ASCII, EBCDIC, or Fieldata); (e.g., an italic character set and a normal character set loaded with one translate table mapping ASCII into one set, and another translate table mapping ASCII into the other set).

The load translate table command is primarily for development use while a new translate/load code table is being tested. After testing it is generally better to place the translate/load code table into a file/element and use the "load character arrangement" or "B" 060 record to load it.

The load translate table command can be at any point in the file, but only affects data records following it. This record does not result in any spacing; only in the loading of a translate/load code table. If the translate/load code table to be loaded is different from that currently selected, the handler/subsystem reselects the originally selected translate/load code table before continuing with the next command.

The translate table is analogous to the 0768/0770/0776 load code buffer.

4.3.10.3. Load Character Arrangement (Subtype 02)

The load character arrangement special print control record has a multi-part format; each section is shown separately. The first section contains the subtype in S1 of Word 0 (see Table 4-22). The remaining sections (see Tables 4-23 through 4-28) do not use S1 of Word 0 of the section. Sections that contain operations of 0 to 2 can be placed in any order with up to eight sections allowed in one record. Sections with operation codes of 3, 4, or 5 must be the only section in the record. If fewer than eight sections are used, and the operation is not 3, 4, or 5, the length of the record passed in the SYMB$ packet must not include any data that is not valid for this record. For example, a seven-section record with four "operation 0" sections followed by three "operation 2" sections must be between 22 and 24 words long. The first section begins in Word 0 of the record and all sections except the last one must contain as many words as shown in the appropriate table.

Table 4-22. Load Character Table Format – First Section (Subtype 02)

00	02	see Tables 4-23 through 4-28	see section descriptions	see section descriptions

Table 4-23. Load Character Table Format – Operation 0 Section

01		0 operation	character generator	font number
02	diskette-id			
03	diskette-id			

Table 4–24. Load Character Table Format – Operation 1 Section

01		1	character generator	unused
02	filename/element name			
03	filename/element name (continued)			
04	filename/element name (continued)			

Table 4–25. Load Character Table Format – Operation 2 Section

01		2 operation	translate table number	unused
02	filename/element name			
03	filename/element name (continued)			
04	filename/element name (continued)			

Table 4–26. Load Character Table Format – Operation 3 Section

01		3 operation	unused	cartridge-id
02	filename/element name			

(continued)

Table 4–26. Load Character Table Format – Operation 3 Section (continued)

03	filename/element name (continued)
04	filename/element name (continued)

Table 4–27. Load Character Table Format – Operation 4 Section

01		4	unused	cartridge-id

Table 4–28. Load Character Table Format – Operation 5 Section

01		5	unused	unused

where:

character generator — This is the character generator number to load. The allowable values are:

0–3

font number — This is the font number to load from diskette. The allowable values are:

0–016

disk–id — This is the volume ID (bytes 5–10 of sector 7, track 0) of the diskette containing the font. If the standard diskette is needed this field can be zeros. The field is a 6–character ID that is left–justified space–filled ASCII in the two–word field.

translate table number	This is the number of the translate table to load. The allowable values are: 0–3
cartridge-id	This is the cartridge-id to use if operation 3 or 4 is specified.
filename/ element name	This is the name of the file (offline systems) or element (online systems) containing the translate table data if the operation is to load the translate table from a file or the font data if the operation is to load a character generator from a file/element. (See 4.3.10.1) Filename/element name are ASCII (case independent).

The load character arrangement control record (02) allows the user to load all of the character generators and translate tables with one instruction, provided all of the information is in a file or on the diskette. It also allows for a return to the system defaults so that processors can leave the file in a standard state for the processors that follow. This record should be the normal means of setting up special fonts and/or translate tables with the load translate table record being used primarily for development of translate tables before they are included in a file or on a diskette.

This record allows the user to change fonts the same as changing bands on the 0770 printer. One danger of this is that the size of the character is fixed and if a 6-lines-per-inch font is loaded and the user prints at 12 lines per inch, the output may be truncated or overlapped. If the system default fonts are loaded, the system assumes responsibility for changing fonts for varying lines per inch provided complete pages are printed with the same lines per inch.

The use of operation code 0 and a diskette-id that is not all zeros implies a desire to use a nonstandard font diskette and results in an operator message to mount the diskette requested. A flag is set to send a message to remount the standard diskette when a return to the default font is requested (for the next file if not earlier).

If any character generator operations are performed with this record, a 2- to 3-second stop of the printer occurs. This record also results in a home paper action and any associated spacing.

4.3.10.4. Load Vertical Format Buffer (Subtype 03)

The load Vertical Format Buffer (VFB) special print control record format is shown in Table 4–29.

Table 4–29. Load Vertical Format Buffer Format (Subtype 03)

00	03	operation	space type	reserved (must be zero)
01	filename/element name			
02	filename/element name (continued)			

(continued)

Table 4-29. Load Vertical Format Buffer Format (Subtype 03) (continued)

03	filename/element name (continued)		
04	page length	top margin	bottom margin
05	line number	lines per inch	skip indicator
06	repeat of above word until the end of the record		

where:

operation — This field is used to determine how the VFB information is to be generated.

- 0 The VFB in its entirety is included in the following data.
- 1 The VFB information is in the file specified by the filename.

space type — This indicates spacing mode to be used.

- 0 This indicates that normal spacing mode is to be used. This mode allows top and bottom margins, truncates spacing to the top of the page when crossing page boundaries, allows headings.
- 1 This indicates that absolute spacing mode is to be used. This mode does not allow top or bottom margins or headings. Spacing over the crease is allowed with the only restriction being a truncation if the space count is greater than a configuration tag (PRSPMX–currently defaulted to 300 lines). Logical line commands and page ejects (spacing equal to 07777) work. The assumed page size for accounting purposes, logical lines, and page ejects is that specified in this record.

filename/element name — This is the name of a file (offline systems) or element (online systems) that contains the description of a VFB. For proper accounting information to be generated at run time the page length and margin information in this record must correspond to that in the file if the file is printed on an offline system. (See 4.3.10.1) The filename is a 12–character ASCII name.

page length — This field contains the length of the VFB.

- 01-03777 These values are taken as actual page lengths in lines to use for setting up the VFB. Some values in this range may be invalid for certain devices and rejected at print time. For example, the 0777 printer requires a VFB of at least 8 inches but not more than 14 inches. It is probable that the offline subsystem will insert some default case for out of range VFBs to allow a degraded printout.
- 07776 Indicates that the page length should not change from its current value.
- 07777 Use the system default value of page size.

top margin — This is the size of the top margin.

- 0-03777 The actual size of the non-printable top margin in lines. This size cannot be larger than the page length minus the bottom margin. This size should correspond to one half inch or more on 0777 printers to ensure good printing. Some devices may truncate this value to 077 or less internally.
- 07776 This value indicates the current top margin size should remain in effect.
- 07777 This value indicates that the default top margin value should be used.

bottom margin — This is the size of the bottom margin.

- 0-03777 The actual size of the non-printable bottom margin in lines. This size cannot be larger than the page length minus the top margin. This size should correspond to one-half inch or more on 0777 printers to ensure good printing. Some devices may truncate this value to 077 or less internally.
- 07776 This value indicates the current bottom margin size should remain in effect.
- 07777 This value indicates that the default bottom margin value should be used.

The following is the data for user-supplied VFBs. If this data is encountered by an older online printer the first entry is checked for the lines per inch to use (12 results in 8) and if a 0770 or 0776 printer the skip indicators may be used to set up the VFB. Remote printers (0768 and older) do not check any data beyond the first lines-per-inch value.

line number — This is the line number on the page that this VFB entry corresponds to. Not all line numbers need be given. The system fills in missing entries by repeating the last entry without the skip code until the line number of the next entry or the line corresponding to the page length specified is reached. All line number entries must be in ascending order.

The value of line number can be from 0 to n where n is the value of page length.

lines–per–inch

This is the vertical density (lpi) associated with this line and all lines until the next specified line.

6	Six lines per inch.
8	Eight lines per inch.
12	Twelve lines per inch.
07776	Use the same value as the previous line or page. If this is the first entry, use the same value as the previous page's first entry. Otherwise, use the value from the previous line.
07777	Use the device default value.

skip indicator

This flag indicates that this line should have a skip code associated with it if it is advantageous for that device. The handler determines the actual skip codes used, the number of skip codes used, and any other handler–dependent skip codes that are to be used.

0	This indicates that no skip code needs to be associated with this line (it is a low frequency use line).
1	This indicates that the programmer believes this line is a good candidate for a skip code (it is a high usage line).

The load vertical format buffer record (03) results in positioning to the home paper position if not already there. Home paper is logically before the top printable line but after all lines on the preceding page.

This is a flexible, expanded variant of the ER PRTCN$ with an M or B (for lpi) control function command. It is interpreted for current printers but only the first five words are used for remote or old onsite printers. The complete record except those having 12 lpi (converted into 8 lpi) and more than one lpi per page are used for 0770 or 0776 printers. User–definable skip indicators lets the user optimize the VFB for the forms being printed. On 0777 printers this can result in a decrease in the number of I/O requests sent to the device, since a well–designed form with the proper VFB does not require any "advance only" commands. Additionally, on 0770 or 0776 printers there is a slight speed advantage when fewer or no "advance only" commands are used. Path length increases do not occur for this means of using the VFB unless the user specifies an improper VFB for the form that results in extra "advance only" commands.

4.3.10.5. Load Graphic Character Modification (Subtype 04)

The load graphic character modification special print control record format is shown in Table 4-30.

Table 4-30. Load Graphic Character Modification Format (Subtype 04)

00	04	operation	character number	data count
01	filename/element name			
02	filename/element name (continued)			
03	filename/element name (continued)			
04	width		height	reserved (must be 0)
05	data or unused			

where:

operation — This field determines the type of operation to perform.

- 0 Indicates that the character modification data is to be loaded into the printer from the data in this record.
- 1 Indicates that the character modification data is to be found in the specified file for loading into the printer.

character number — This is the index into the character generator for the character to be modified. The allowable values for the 0777 printer are:

0-0376

data count — This is the exact number of significant bytes in the data area of this record. The allowable values for the 0777 printer are:

0-60

width — This is the number of dots wide the character matrix is.

01-024 The allowable values for the 0777 printer. On 0777 printers the vertical column associated with the width specified contains the end of character code.

07777 This value indicates that a maximum width for the device is to be used without an end-of-character indication.

height — This is the number of dots high of the given character matrix. If the device allows more dots than this, the extra height is filled as if it were non-printing.

01-030 The allowable values for the 0777 printer.

07777 Use the maximum height for this printer.

filename/element name — This is the name of the file (offline systems) or element (online systems) to use if the operation requires a file/element. The filename is in ASCII and has a maximum length of 12 characters, left-justified space-filled. (See 4.3.10.1)

data — The data words are a series of bytes that map into dot positions in the character matrix. These bytes are 9-bit bytes (8 significant) that are packed four bytes to a word. Each bit in the byte maps into one dot position of the printed character. Each vertical column of the character starts on a byte boundary. Therefore, if a height of 20 characters is specified there are three bytes given for each vertical column. In the case of the 0777 printer, extra bits that are needed (more vertical bits and control information) are added by the subsystem or handler.

The bytes specified in the data area are a subset of those required by the 0777 printer, but the subset allows full user definition of the character.

The load graphic character modification command is used to modify one or more characters for the user's use. A standard font that includes the characters desired should be used or a font on a separate diskette that can then be loaded into the printer should be created. This feature is useful when only a few characters need to be changed or the user desires to check the appearance of certain characters.

This command results in the printer being positioned at the home paper position if not already there and results in a delay of approximately two to three seconds, since the printer must come to a complete halt while the operation is performed.

4.3.10.6. Select Electronic Forms (Subtype 05)

The format for the special print control record to select electronic forms is shown in Table 4-31.

Table 4-31. Select Electronic Forms Command Format (Subtype 05)

00	05	operation	start copy	number of copies
01	filename/element name			
02	filename/element name (continued)			
03	filename/element name (continued)			

where:

operation — This is the type of electronic forms operation to use.

- 0 - Only use electronic forms on the following base page.
- 1 Use this electronic form on all the following base pages until turned off or until the form is changed.
- 2 Turn off any prior electronic forms.

start copy — This is the number of the first copy of a page that is to get the electronic forms. For this command to be effective a "load copy number" command with at least the starting copy number plus the number of copies as a copy number must be performed. The allowable values for 0777 printers are:

1-255

number of copies — This is the number of copies of each page to which to apply the electronic forms. The allowable values for 0777 printers are:

1-255

This value plus the starting copy number must be less than 255.

filename/element name — This is a 12-character ASCII name of a file (offline systems) or element (online systems) that contains the electronic forms specification.

The select electronic forms record (05) merges the data in the named file, which is used to create an electronic form, with the data records from the file. This is not a true electronic form in that it uses special characters to generate the forms. Character locations that contain forms information cannot contain any print data other than spaces or underlines or the results for those locations are unpredictable. This record results in positioning at home paper if not already at home paper.

4.3.10.7. Load Copy Modification (Subtype 06)

The load copy modification special print control record format is shown in Table 4-32.

Table 4-32. Load Copy Modification Format (Subtype 06)

00	06	operation	unused	data count
01	start copy		number of copies	start line
02	number of lines		start column	dup factor
03	data			

where:

operation — This is the type of copy modification operation to perform.

- 0 This data only affects the next base page of data that is sent.
- 1 This data should be used with all future base pages until directed to stop.
- 2 Stops all copy modifications that are being performed.

data count — This is the exact number of bytes of copy modification data in this record. Does not include Words 1 and 2. The allowable values for 0777 printers are:

1-204

start copy — This is the number of the first page to which this copy modification data applies. The allowable values for 0777 printers are:

1-255

number of copies — This is the number of copies of this page to which to apply this data. The allowable values for 0777 printers are:

1-255

This value added to the start copy number must be less than 257.

start line — This is the first line to which this data applies. The allowable values for 0777 printers are:

1–168

number of lines — This is the number of print lines this data affects. The allowable values for 0777 printers are:

1–168

This value plus the start line value must be less than 168.

start column — This is the starting column number for positioning this data. The allowable values for 0777 printers are:

1–204

dup factor — This is the number of repetitions of the data to send for each line. The allowable values for 0777 printers are:

1–204

This value times the byte count plus the start column cannot exceed 204.

data — This is character data that is to overlay the selected portions of the data for the page. This data is in the format described by S6 of the control word and should use the translate table indicated in S5 of the control word.

The load copy modification record (06) allows the user to send a page once, tell the printer to print several copies (via the "load copy number" command), and then modify selected portions of selected copies.

This record results in positioning at home paper if not already at home paper.

4.3.10.8. Allow/Inhibit Data Check (Subtype 07)

The allow/inhibit data check special print control record format is shown in Table 4–33.

Table 4–33. Allow/Inhibit Data Check Format (Subtype 07)

00	07	operation	reserved	reserved

where:

operation — determines the actual handling of the record.

0 If a data check occurs, terminates the file immediately with an error message. "DATA CHECK ERROR - FILE TERMINATED" Implies that data checks are to be enabled.

1 If a data check occurs, inhibits data checks, prints the rest of the file, and prints a message at the end of the file that at least one data check occurred. ("ATLEAST 1 DATA CHECK OCCURRED WHILE OUTPUTING FILE *file*"). This implies an enabling of data checks until a data check occurs or a request is submitted to inhibit data checks.

2 This inhibits data checks.

The allow/inhibit data check control record (07) allows a user to turn on the "allow data check" feature for the remainder of the file, until the first error occurs, or until data checks are inhibited. The default is that data checks are inhibited unless enabled by this command.

This is implemented for 0770 and 0776 printers. This record does not cause any spacing to be done.

4.3.10.9. Special Forms (Subtype 010)

The special print control format to receive special forms (write to console) is shown in Table 4-34.

Table 4-34. Write to Console Format - Special Forms (Subtype 010)

00	010	reserved
01	form-id	
02	form-id	
03	form-id	

where:

form-id This is the local identification for the form. It consists of one, two, or three words of ASCII data (9-bit bytes). If only one or two words are given, the last words are assumed to be spaces.

The special forms record is a means of informing the operator about the need for special forms. It results in a type and read operator message with the form-id inserted in an appropriate place. Once this command is received, a flag is set indicating that special forms are mounted. This results in a restore message before printing the next file. The restore message is not sent if the printer is a 0777 subsystem and the next file is started without interruption and contains a similar command with the identical form-id before any print data records.

This record results in the paper being spaced to the home paper position if not already there.

If the print is to go to a NTR batch site that is operating under OS-3, then only an 8-character form-id should be used.

This capability is implemented for all printers.

4.3.10.10. Load Copy Number (Subtype 011)

The format for the load copy number special print control command is shown in Table 4-35.

Table 4-35. Load Copy Number Command Format (Subtype 011)

00	011	operation	reserved	number of copies

where:

operation — This is the type of operation to perform.

- 0 — Only use the copy number for the next base page.
- 1 — Use the copy number for all subsequent base pages until a new "load copy number" command is received.

number of copies — This is the number of copies desired.

- 0 — Turns off the multi-copy function.
- 1-255 — The allowable range of copy numbers for the 0777 printer.

The load copy number command allows common data for several pages to be transmitted once to the printer instead of once for each page. It is required if a "load copy modification" or "load flash number" command is used.

This command does not result in any spacing but does have certain restrictions regarding placement and use with other commands (see 4.3.10.12). This command results in home paper if not already positioned at home paper.

4.3.10.11. Load Flash Number (Subtype 012)

The format for the load flash number special forms command is shown in Table 4-36.

Table 4-36. Load Flash Number Command Format (Subtype 012)

00	012	operation	reserved	number copies
01	flash-id			
02	flash-id			
03	flash-id			

where:

operation — This determines the type of control desired.

- 0 — Only the following base page is to be flashed.
- 1 — All subsequent pages are to be flashed until the next "load copy number" or "load flash number" command.

number of copies — This is number of copies to flash.

- 0 — Turns off any flashing due to previous records.
- 1-255 — The number of copies to flash.

flash-id — This is the name of the negative to use for flashing. The name is in ASCII. If this field is present, an operator message is sent to mount this negative in the printer. The absence of this field implies that the negative is already mounted. This command also implies that a message is not printed at the end of the file for removal of the negative. This does not create problems since the subsystem does not flash the negative unless commanded to and if a subsequent file requires a forms flash, an operator message is sent before starting to flash the negative (if the user wanted to).

The load flash number command is used in conjunction with the "load copy number" and "load copy modification" commands to place constant information from a negative onto the page. It has special restrictions as to the need for other commands to be active and positioning in the file (see 4.3.10.12). This command results in home paper positioning if not already at home paper.

4.3.10.12. Use of Special Print Control Record Commands

The "load copy number", "load flash number", "load copy modification", and "electronic forms" commands are used together. If any one is to be used on a page, it must be sent prior to the base page data. If more than one of these commands are to be used on the same page, then all of the commands to be used must be contiguous with no spacing or data record intermixed. Any one of these commands results in the termination of the data for the last base page and a space to the home paper position.

If a "load flash number", "load copy modification", or "electronic forms" command is used, a "load copy number" command must either be used or a previous one must still be in effect.

A base page is the data records representing the common data for a number of pages that are produced by a "load copy number" command.

4.3.11. PCHCN$ Control Records (070)

The punch control record format is shown in Table 4-37.

Table 4-37. PCHCN$ and READ$ Data Record Format (070)

00	070	length	reserved	code type
00	data word 0			
01	data word 1			

The ER PCHCN$ control record consists of a control word with a 070 in S1 and a special control record in the following several words. Section 4.3.11 discusses these special control records. The special control word has the length of the special control record in S2 and the file type (1 for ASCII or 0 for Fieldata) in S6. The record consists of one or more punch control functions to alter the punching of the file and has a maximum length of 63 words.

The ER PCHCN$ punch control functions are interpreted separately and acted on as if they were in separate records. Separate statements in the same record are separated by periods. A space period space terminates the record scan. Therefore, the remainder of the record can be used as a comment field. Note that a period immediately following a nonspace character does not terminate the record even if followed by spaces.

- READ$ Usage of the 070 Record:

When the input symbionts encounter an @COL record, an 070 record with the appropriate C,*code* record is placed into the READ$ file as a type change record.

4.3.11.1. Special Control Functions for PCHCN$

The special control functions for PCHCN$ are a subset of those functions used for the ER PRTCN$ control records (060). In addition to the card code (C) function discussed in 4.3.11.2, the special control functions for PCHCN$ are:

- Special forms (S) – See 4.3.9.4.
- Skip break (R) – See 4.3.9.8.
- Width (W) – See 4.3.9.9.
- User (U) – See 4.3.9.10.

4.3.11.2. Card Code (C) Control Function

Format:

C,*code*

where *code* can be one of the following values:

B	Punch the following images in column binary.
E	Punch the following images in the last character mode selected by an 070 record. (End of binary data.)
ASC	Punch the following images in ASCII (Hollerith–A card punch format).
9000	Punch the following images in EBCDIC (9000 version).
1100	Punch the following images in Fieldata (Hollerith–H card punch format).

NOTE: "ASCII" and "Fieldata" describe the pattern of holes punched in the cards. Do not confuse them with the internal representation of the characters (the familiar meaning of these terms).

4.3.12. End–of–File Control Record (077)

The end–of–file control record format is shown in Table 4–38.

Table 4–38. EOF Data Record Format (077)

00	077	data count	reserved
00	data word 0		

(continued)

Table 4-38. EOF Data Record Format (077) (continued)

01	data word 1
n	data word n

The end-of-file control record is used to indicate the end of actual file data. S1 of the control word contains 077. Any words in the file that follow this record are disregarded. Normal symbiont complex practice is to fill the last 224-word block to the end with end-of-file control records since the symbionts require the last block to be at least 224 words long. Tapes must have the last block equal to 224 or 1792 (0777 only) words in length.

PCIOS and the System Data Format Output routine (SDFO) normally only place one end-of-file record in the file. PCIOS places it as the last word of the last sector. The SDFO routine places it immediately following the last record.

5. Relocatable Binary (RB) Format

A relocatable element is the output of the language processors, e.g., FTN, ACOB, and MASM, and sometimes the Series 1100 Collector (via a @MAP,R control statement). Relocatable elements are in relocatable binary (RB) format and are used as input to a collection to produce an executable program (i.e., absolute element).

RB format is optimized, as far as storage space and relocation efficiency, for the instructions and data to be coded under control of location counters one and two. Some compilers use, almost entirely, location counter one for instructions and location counter two for data of the generated code.

The relocatable binary element is divided into two parts called the preamble and the text. The preamble contains information concerning the external properties of the element and the text contains the generated instructions plus the relocation information.

5.1. The Preamble

The preamble is composed of five tables located after the text of the relocatable element. These tables, in the order given, are:

Base Table

Location Counter Table

Undefined Symbol Table

Entry Point Table

Control Information (INFO) Table

Except for the base table, only those tables necessary to a particular element are included in the preamble. The base table is always included. The preamble is a variable-length table.

5.1.1. Base Table

The format for the base table, always included in the preamble, is shown in Table 5–1.

Table 5–1. Base Table Format

00	index to location counter table	item count of location counter table
01	index to undefined symbol table	item count of undefined symbol table
02	index to entry point table	item count of entry point table
03	index to control information table	item count of control information table

The indices in the left half of the entry are relative to word zero of the base table. This table is fixed in length. If one of the other four tables is not present, its item count is cleared to zero.

5.1.2. Location Counter Table

The location counter table format is shown in Table 5–2.

Table 5–2. Location Counter Table Format

00	K-bit count	location counter 0 length of counter
01	minimum data area (D-bank Address)	location counter 1 length of counter
≈		≈
n–1		location counter *n–1* length of counter

(continued)

Table 5-2. Location Counter Table Format (continued)

n		location counter *n* length of counter

where *n* must be less than 512.

The K-bit count is the number of bits needed to contain:

a. the number of undefined symbols in the element, or

b. the largest location counter number, whichever is greater

This must be the same value supplied to the relocatable output routine (ROR).

The minimum address assigned to the data area of this element is normally zero. However, for some programs, the minimum address is specified by the compiler, generating INFO Group 3.

The location counter table is a variable-length table with a maximum length of 512. The location in the table is the counter number. The length of the table is equal to the highest numbered location counter used in the element plus one.

For example, assume that counters 1, 3, 4, and 7 are used, and the number of words reserved or generated are 20, 30, 5, and 3 under the respective counters. In addition, there is an entry point under the void location counter 6. This location counter table example is shown in Table 5-3.

Table 5-3. Location Counter Table Example

00	K-bit count	0
01	0	20
02	0400000	0
03	0	30
04	0	5

(continued)

Table 5–3. Location Counter Table Example (continued)

05	0400000	0
06	0	0
07	0	3

If H1 = 0400000 and H2 = 0, the location counter is nonexistent. Otherwise, a word of all zeros indicates a void location counter. The use of void location counters should be avoided since it makes the Collector less efficient. However, location counters 0 and 1 cannot be marked nonexistent.

If H2 = 1, the location counter has a length of zero.

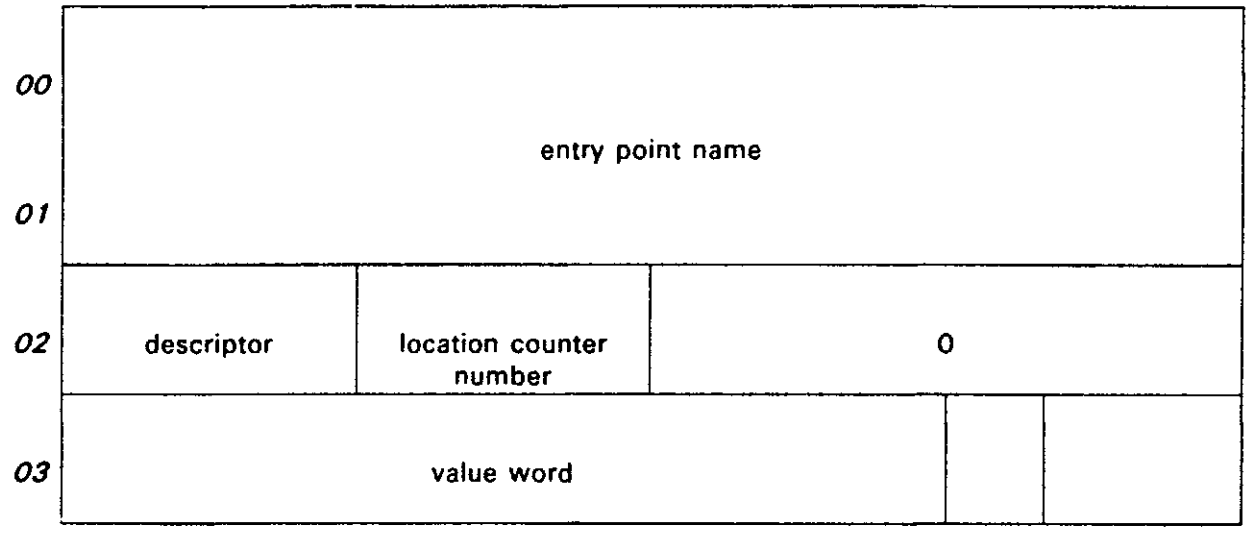
5.1.3. Undefined Symbol Table

The undefined symbol table is a list of all the symbols that are not defined within the element. The undefined symbols may be up to twelve characters in length; each entry in the table is two words. The text, in place of the undefined symbol, contains indicators which link these specific references to the undefined symbol table. The value of the indicator is the order of the symbol in the table beginning with zero. The indicator value multiplied by two gives the index into the table of the first word of the undefined symbol. There may be up to 1024 undefined symbols in an element. The entries are left-justified and space-filled Fieldata. Entries in the undefined symbol table need not be in any particular order (e.g., lexically ordered).

5.1.4. Entry Point Table

The entry point table format is shown in Table 5–4.

Table 5–4. Entry Point Table Format



Each item is four words in length. The first two words contain the name of the entry point (LJSF Fieldata). The third word contains a descriptor, and the fourth word is a value word. Bit 27 of the third word is the type flag. If bit 27 is one, the entry point is absolute and has the value of the value word. If bit 27 is zero, the value word is relative to the location counter in bits 18 through 26. Entries in the entry point table need not be in any particular order.

Any number of entry points may be given.

5.1.5. Control Information (INFO) Table

Listed below are the Control Information Table (CIT) entries recognized by the Collector.

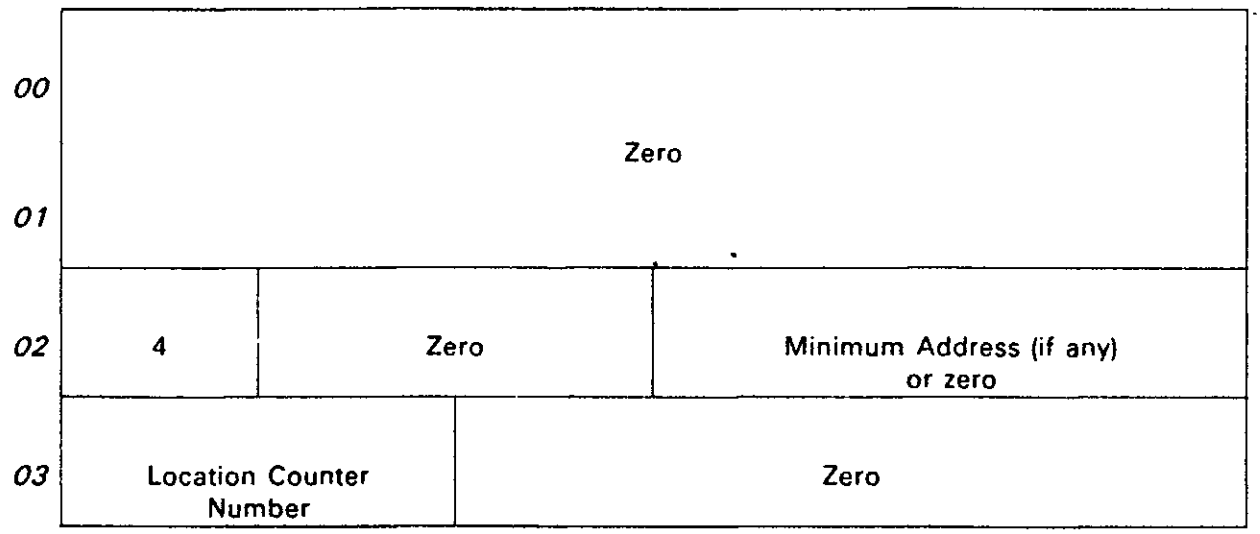
INFO Group 2 Entry (Named Common Block):

00	Common Block Name		
01			
02	2	Zero	Minimum Address (if any) or zero
03	Location Counter Number	Zero	

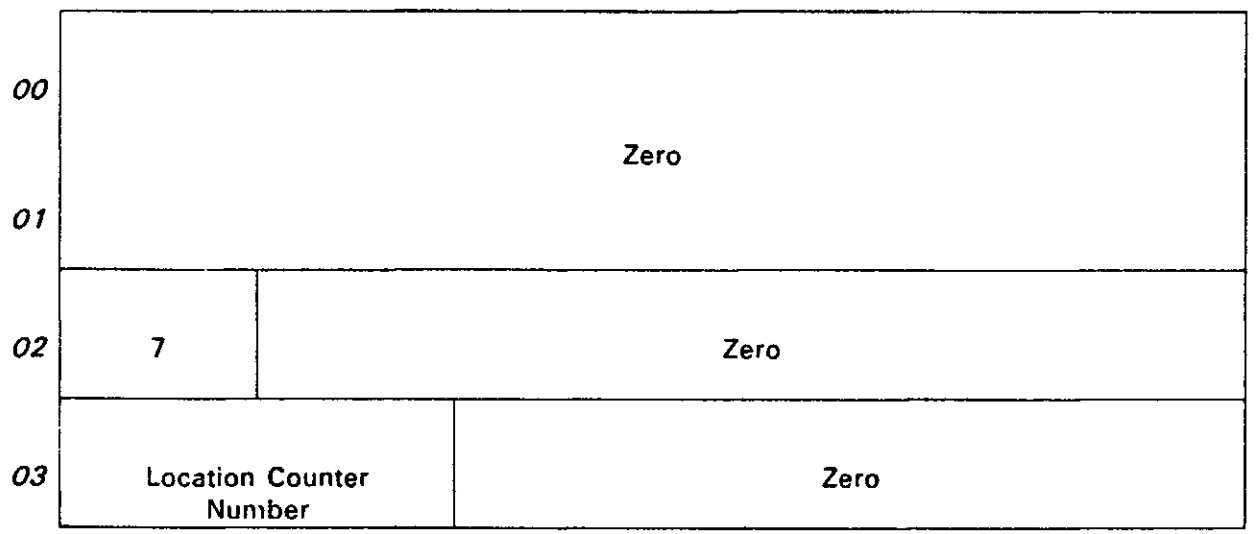
INFO Group 3 Entry (Minimum Address Specification):

00	Zero		
01			
02	3	Zero	Minimum Address (if any) for D–bank or zero
03	Zero		Minimum Address (if any) for I–bank or zero

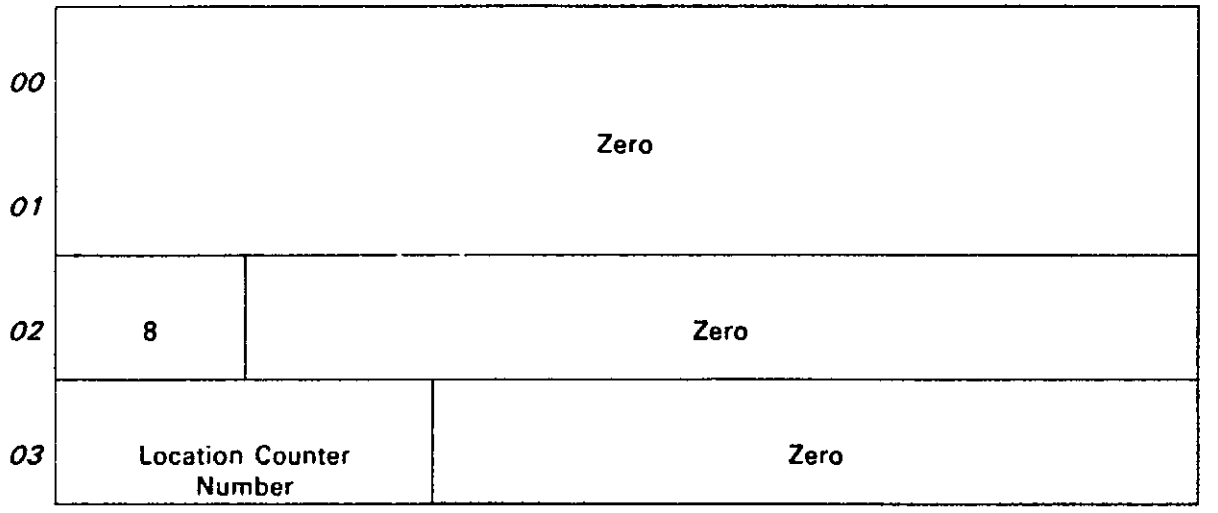
INFO Group 4 Entry (Blank Common Block):



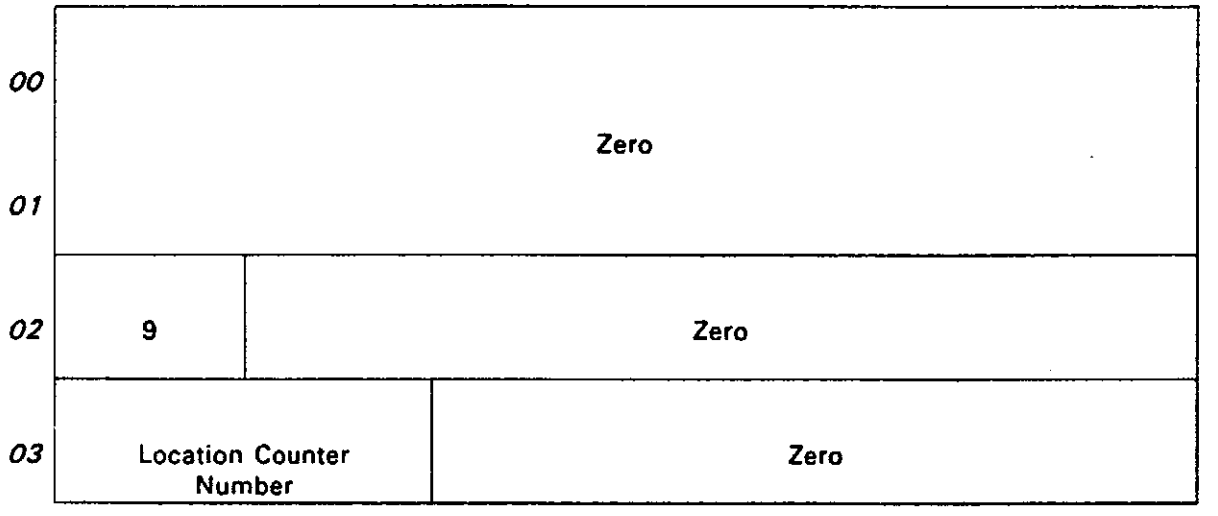
INFO Group 7 Entry (Even start address for Location Counter):



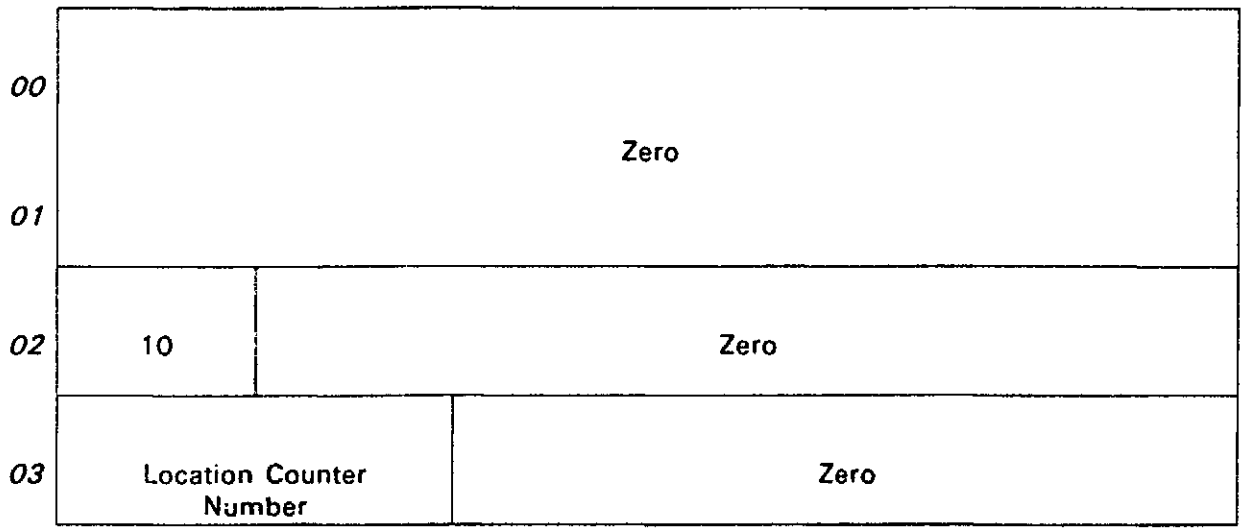
INFO Group 8 Entry (Diagnostic Information):



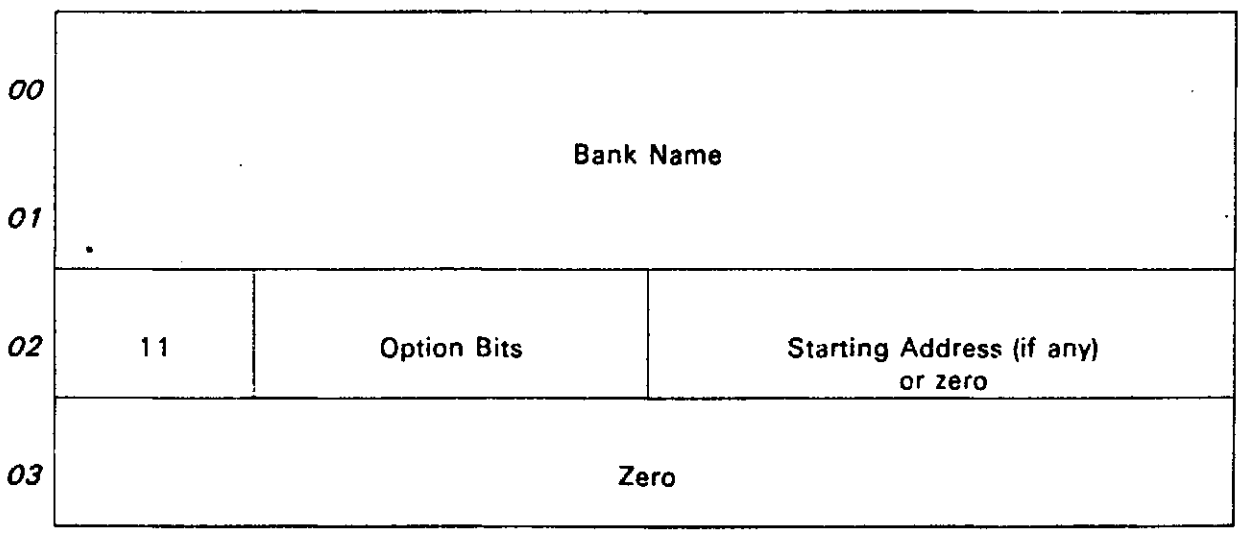
INFO Group 9 Entry (Read-only Location Counter):



INFO Group 10 Entry (Extended Mode Location Counter):



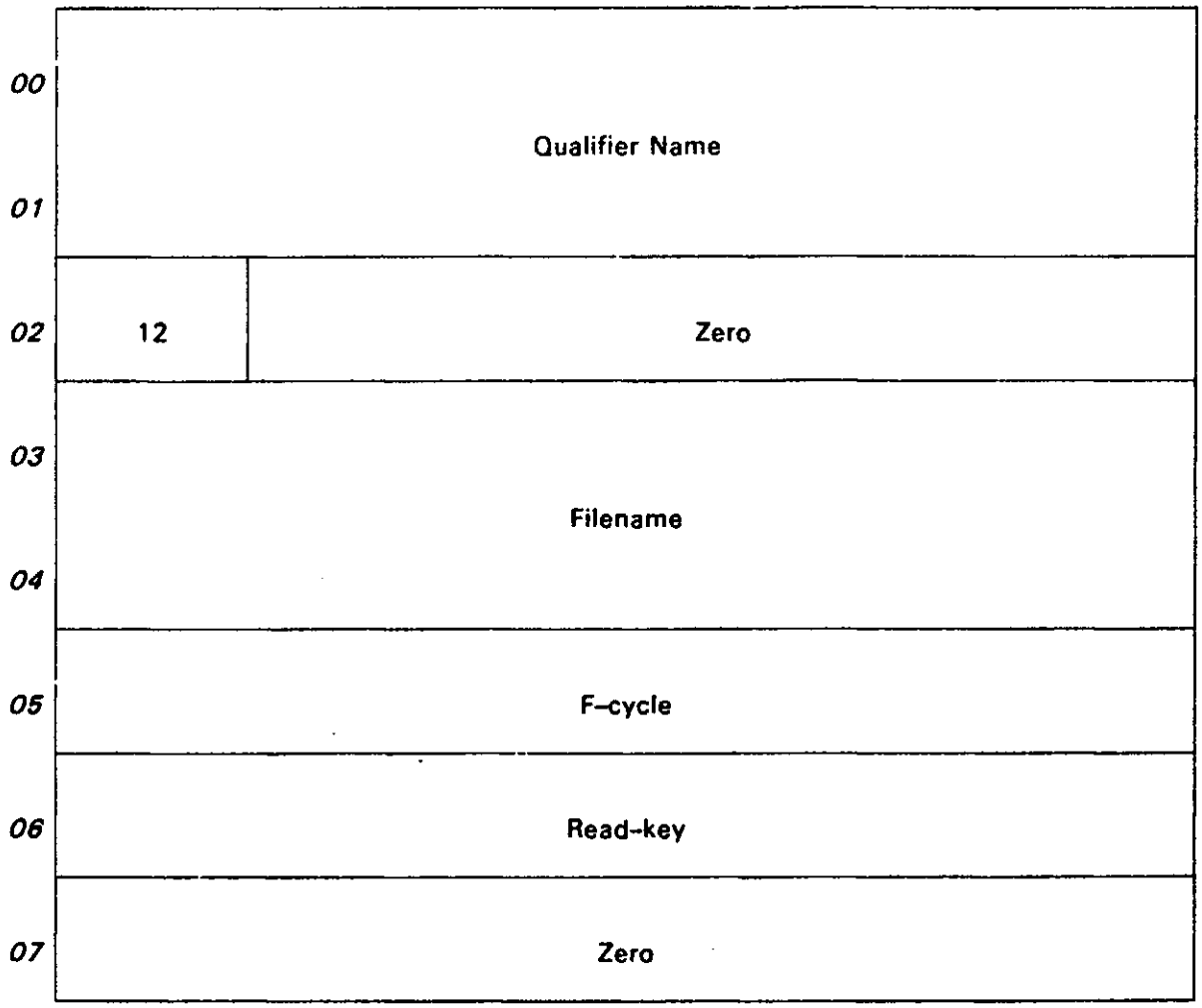
INFO Group 11 Entry (Void Bank):



Option bits:

Bit 18 - S option (shared bank)
Bit 19 - D option (dynamic bank)
Bit 20 - I-bank

INFO Group 12 Entry (LIB Entry):



The Collector allows a location counter to be defined for only one of the groups 2, 4, or 8. The INFO Groups 1, 5, and 6 are also handled by MASM however they do not produce CIT entries but pass their information to the Collector by different means.

5.2. The Text

In general, compilers and assemblers need not be aware of the text format. The system relocatable output routine (ROR) formats the text when called by the compilers or assemblers.

The text is broken into blocks which in turn are broken down into word groups. The word groups are broken down into text words and relocation information related to the text words. The block size is fixed by the relocatable output routine (ROR) at 14 words.

The relocation information is optimized for the most heavily expected use of two location counters in a word group. These location counters are called normal counter one and normal counter two, or for short, NC1 and NC2. NC2 is always location counter two. NC1 is the location counter controlling the text words of this word group except when the text words are controlled by location counter two. In this case, NC1 is location counter one.

The notation (NC1) and (NC2) is used to indicate the current assignment of NC1 and NC2 respectively. The current assignment of a location counter is the absolute starting address of that counter as assigned by the Collector.

Within the relocation information, fields refer to location counters and undefined symbols. These fields are K bits in length. The compilers and assemblers may specify the value of K. If the value of K is not specified, the fields are assumed to be ten bits in length. The value of K supplied to the relocatable output routine must be in the left half of the first word of the location counter table in the preamble.

In all discussion concerning the text, the bit positions of the computer word are numbered from 00 to 35 from right to left. Any connected portion of a word, from bit M to bit N (where M > N) may be relocated mod $2^{(M-N+1)}-1$. Relocation specifications may be specified in an arbitrary manner.

The possible relocation specifications are:

- relocation by any of the 512 location counters,
- relocation by any of the 1024 undefined symbols,
- addition or subtraction of relocation increments, or
- repeated addition and/or subtraction of relocation increments.

Special tests are made by the Collector to flag any possible field overflow when the field could be used as an address. The fields include bits 00–15, 00–17, 18–33, and 18–35.

5.2.1. Word Groups

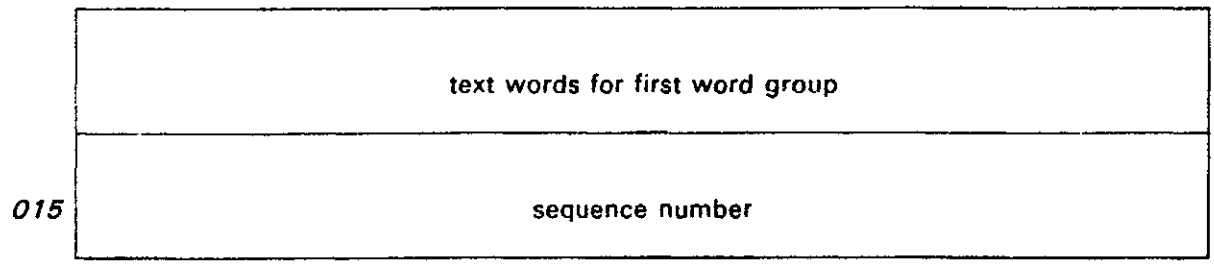
Word groups represent text words that are allocated to sequential memory cells. Each word group is made up of two parts: the word relocation information and the text words themselves. The relocation information for the word groups starts at the beginning of the block and the text words are at the end of the block as shown in Table 5–5.

Table 5–5. Relocation Information and Text Words

relocation for first word group
relocation information for second word group
≈ ≈
text words for second word group

(continued)

Table 5-5. Relocation Information and Text Words (continued)



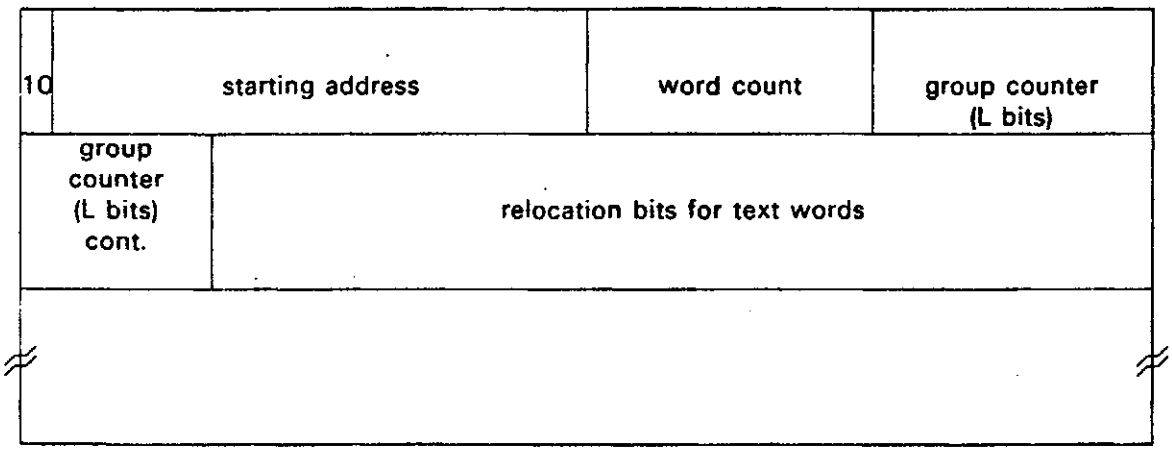
The text words are in inverse order in the block from the way they are allocated to memory cells.

The last word in each block is a sequence number. Bits 35–24 are two Fieldata characters. The first begins with A and cycles through the entire Fieldata character set. The second is always a letter. Bits 23–0 are always zero.

The relocation information in any word group is in the form of a "stream of bits." The length of the stream is variable, but the first bit of a stream is always the left–most bit of a word. See Table 5–6.

The relocation information for each word group includes fixed– and variable– length fields.

Table 5-6. Relocation Information Bit Stream



The first bit of the stream is always the left–most bit of a word.

Bits 35 and 34 contain the binary value 10.

Bits 33 through 18 of the stream contain the program starting address of the first text word in the word group relative to the start of the group counter.

Bits 17 through 9 contain the number of text words in the word group.

The group counter number starts in bit 8 and may vary in length from 2 to K + 2 bits (and thus may overflow into the next word). (See 5.2.2.)

The relocation bits for the text words start in the bit following the group counter. (See 5.2.3.)

The program transfer address (start address) is represented by a bit stream whose first two bits are 11. The word count for this bit stream is zero.

5.2.2. Group Counter

The group counter is the location counter under which the text words are to be placed. During collection, the starting address of this location counter is added to the word group starting address to determine the absolute program address of the text words. The group counter is determined by the L bits following the word count field.

The group counter also determines what location counter is assigned to NC1.

Group Counter	NC1	NC2
0	0	2
1	1	2
2	1	2
X where X>2	X	2

5.2.3. Relocation Bits

The stream of relocation bits governs the relocation of the text words. The possible bit combinations and their meanings are given below.

The bits in the text words are numbered 00 to 35 from right to left.

Figure 5-1 shows how the modification is accomplished.

The processors supply the relocatable output routine with a 19-bit signed number for the following relocated fields:

bits 00-15

00-17

18-33

18-35

Bit 19 is the sign of the field. The text word contains either 16 or 18 bits of the field and the modification bits contain the most significant bits of the field (either 3 bits or 1 bit).

The modification bits are grouped for logical functions. The field to be modified is selected and then the modification to take place is determined. Other bits determine when the modification bits apply to another word, another field, or the same field.

The first information in the stream of bits is used to determine the field to be relocated if relocation is desired. If the first bit is one, the relocation field is the right address (bits 00-15), and three more bits in the stream are retrieved to complete the 19-bit number. If zero, the next bit is examined. If

the next bit is also zero, no relocation occurs in any part of the text word. This bit set to one indicates the field is determined by the next two bits. The bits set to 00 indicate the left address field (18–33); 01 indicates the right-half field (00–17); 10 indicates the left-half field (18–35); and 11 indicates that the field is defined by the next twelve bits. The first six bits of the twelve bits give the left-most bit number of the field and the last six bits give the right-most bit number of the field. For left address fields, three more bits are in the bit stream to complete the 19-bit value of the field. For right- and left-half fields, the next bit in the stream is the sign bit of the field.

The possible bit configurations to select a field are then:

00	No relocation of any field
1SXX	Right address field with SXX the most significant bits of the field
0111ZZZZZZ ZZZZZZ	Field is specified by 12 bits and is not RH, LH, RA, or LA
0100SXX	LA and field extension SXX
0101S	RH and field extension S
0110S	LH and field extension S

The bit following the field specification determines if the relocation is done by subtraction or addition: if one, it is done by subtraction, if two, it is done by addition.

The next two bits determine the relocation value to add to or subtract from the field. They are:

00	Modify (add or subtract) by the value of NC1
01	Modify by the value of NC2
10	Modify by the value of the undefined symbol in the next K bits
11	Modify by the value of the location counter in the next K bits

If the next bit is zero, the relocation of the text word is complete. The following bits apply to the next text word in the same word group.

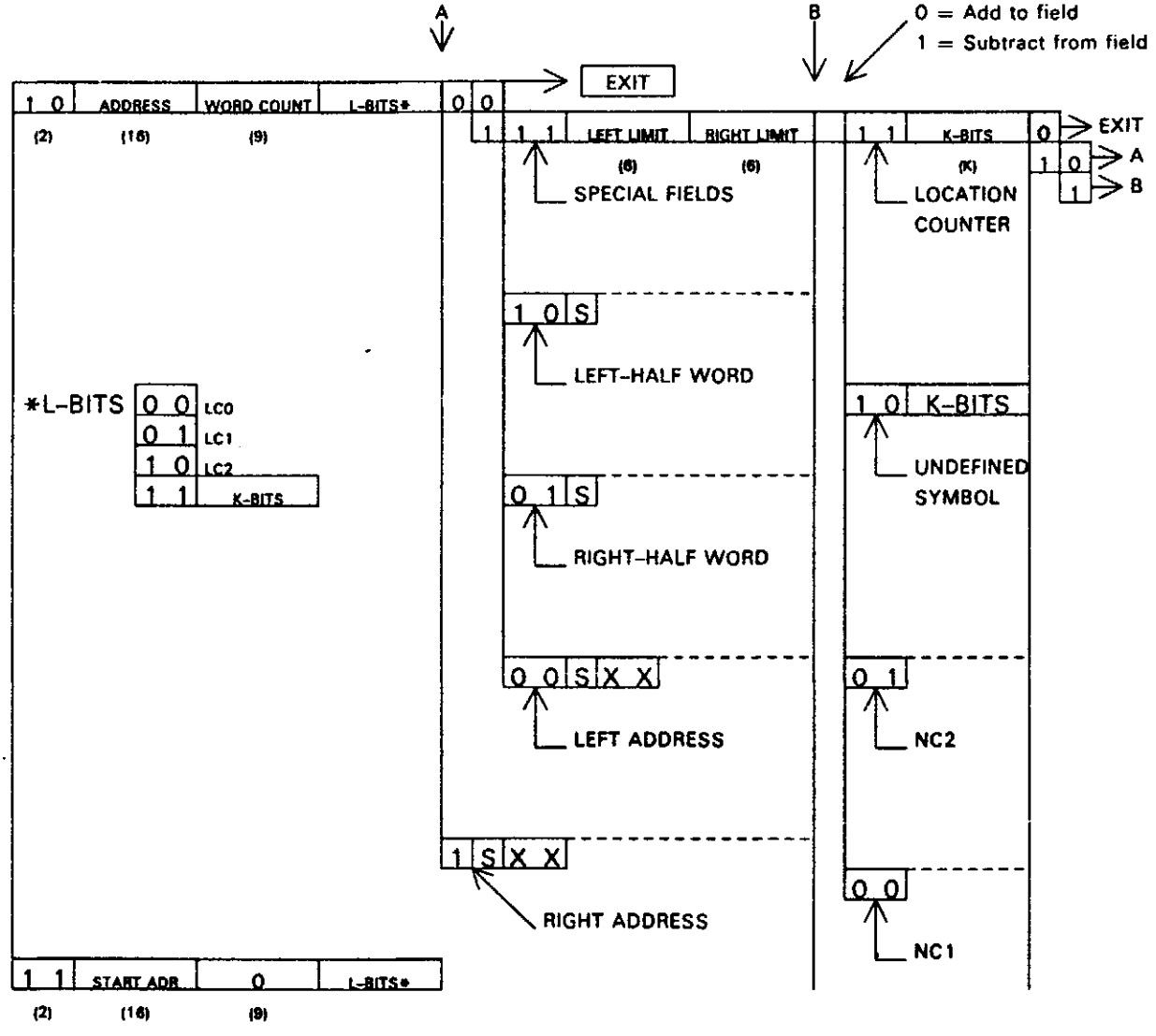
Some possible bit streams for one text word are:

1SXX0010	Add the value of NC2 to the right address.
1SXX01100000001000	Add the value of location counter 4 to the right address (K=10).
1SXX000100110S11000000000100	Add NC1 to right address and subtract undefined symbol 2 from left half (K=10).

When a field is extracted from the text word, the most significant bit of the field is used as the sign bit. RH, LH, RA, and LA fields are the exception. Bit 19 is the sign bit for these fields and is a part of the bit stream as indicated.

5.2.4. Relocation Field Selection

Figure 5-1, Relocation Bit Stream, diagrams the necessary information for relocation field selection.



WORD COUNT- Number of text words, maximum of 14

S - Field sign

X X - Bits 17 and 16 of address field

A - More modification, different field

B - More modification, same field

K-BITS - Fixed field for each element, width (K) as given in Word 0 of location Counter Tabl

EXIT - No more relocation for this word; next word in group starts at A.

Figure 5-1. Relocation Bit Stream

6. Absolute Element Format

An absolute element contains a complete program in binary form suitable for execution by the Executive. Such elements normally occur as output from a collection of relocatable elements, with all necessary linkages and relocation performed.

The Header Table is always at the beginning of the element and is 28 words in length. It is followed by the Bank Load Table (BLT). This information is used by the Series 1100 Executive to load program banks.

Following the BLT may be a common bank list. This is used by the Executive to keep records of common bank usage.

Following the above tables are the overlay segments. The main segment follows the overlay segments. Intermixed with a segment's instructions and data are load control groups. Each 28-word load control specifies the relative storage address of the blocks of instructions and data that follow. All load control groups start at a sector boundary when the program resides on sector-formatted mass storage.

Normally a program's instructions are contained in logical structures called I-Banks while the data is contained in D-Banks. One bank of a program (either an I-Bank or D-Bank) is designated as the control bank. For segmented programs, the table used in loading the segments is included in the main segment of the program's control bank.

A table of entry points to the program, a table of undefined references from the program, and a list of common data areas are also included in the main segment of the control bank when entry points are specified by DEF directives or external references are specified by REF directives in the Collector source language.

The various tables used by diagnostic routines follow the main segment. These tables are:

- Static Diagnostic Pointer Table and Static Diagnostic Text
- Segment Name Table
- Element Name Table
- Bank Name Table
- Location Counter Table

- Entry Point Name Table
- Absolute Value Definition Table

Any relocatable segments (RSEGs) in the program are output first. The overlay segments are output in the same order as specified in the source language with the main segment being output last. Each segment is output by portion within a bank in order of ascending Bank Descriptor Indexes (BDIs).

Banks are assigned ascending BDIs as follows:

- Z option banks
- S option banks
- non–initially based dynamic banks
- initially based dynamic banks
- static banks
- control bank
- INFO–11 banks

6.1. Absolute Element Format

The absolute element format is shown in Table 6–1.

Table 6–1. Absolute Element Format

Header Table
Bank Load Table --- Common Bank List (if present)
Text of Overlay Segments
Text of Main Segments (all but the control bank) --- Segment Load Table (SLT$) (if present) --- Entry Point Table (ENTRY$) (if present) --- External Reference Table (XREF$) (if present)

Table 6-1. Absolute Element Format (continued)

Indirect Load Table (if present) - - - - - - - - - - Text of Main Segment of Control Bank
Static Diagnostic Pointer Table (present if any INFO Group 8 in collection)
Static Diagnostic Text (present if any INFO Group 8 in collection)
Segment Name Table (SNT) (present unless Z option used on @MAP call)
Element Name Table (ENT) (present unless Z option used on @MAP call)
Bank Name Table (BNT) (present unless Z option used on @MAP call)
Location Counter Table (LCT) (present unless Z option used on @MAP call)
Entry Point Name Table (EPNT) (present unless Z option used on @MAP call)
Absolute Value Definition Table (ABSV) (present unless Z option used on @MAP call)

NOTE: Solid lines denote sector boundaries.

6.2. Header Table Format

The header table format is shown in Table 6-2.

Table 6-2. Header Table Format

00	sentinel (0444444444444)						
01	E	0	BDI-initial PSRMI (BDR 0)	E	0	BDI-initial PSRMD (BDR 2)	
02	E	0	BDI-initial PSRUI (BDR 1)	E	0	BDI-initial PSRUD (BDR 3)	
03	0						
04	E	0	BDI-control bank	Program Start Address			
05	Sector Address of Main Segment ACWs for Initially Loaded Banks						
06	Number of 4 word SLT entries			SLT word length			
07	B	0					
010	0						
011	0						
012	0						
013	0						
014	0						
015	time and date of absolute element creation (reversed TDATE$ format)						
016	Fieldata SYS$*RLIB$ ID						
017	options on @MAP in master bit notation						
020	sector address of diagnostic tables						
021	Static Diagnostic Pointer Table word length			control bank SLT$ address			
022	number of ENTRY$ entries			control bank ENTRY$ address			
023	number of COMMN$ entries			control bank COMMN$ address			
024	number of XREF$ entries			control bank XREF$ address			
025	number of Indirect Load Table entries			control bank IDL table address			
026	word length of LCT			word length of BNT			
027	word length of SNT			word length of ENT			
030	word length of EPNT			word length of static diagnostic text including pointer table			

Table 6–2. Header Table Format (continued)

031	word length of ABSV		word length of BLT
032	C	total mass storage word length of element	
033	0		

NOTE: All sector addresses are relative to the Header Table sector address.

When tables are not included, the corresponding entries in the Header Table are cleared to zero. Abbreviations used are as follows:

E – Set if the BDI is for the EXEC BDT

B – Set if TYPE BLOCKSIZE64 was specified

SLT – Segment Load Table

LCT – Location Counter Table

BNT – Bank Name Table

SNT – Segment Name Table

C – Set by partial checkpoint

EPNT – Entry Point Name Table

ABSV – Absolute Value Table

BLT – Bank Load Table

ENT – Element Name Table

6.3. Bank Load Table (BLT) Format

The Bank Load Table (BLT), actually the Preliminary Bank Descriptor Table, format is shown in Table 6–3.

Table 6–3. Bank Load Table (BLT) Format

BDI	Sector address of bank's main segment ACWs

Table 6-3. Bank Load Table (BLT) Format (continued)

type	preference	block count	SLR lower

Type bits are set as follows:

	Specified by these options on IBANK or DBANK directives:
Bit 35 - Bank is dynamic	D
Bit 34 - Bank is a D-bank	DBANK directive
Bit 33 - Write protect set for bank	R
Bit 32 - Bank is a guaranteed entry bank	G
Bit 31 - Bank is an S option shared bank	S
Bit 30 - Bank requires test and set queuing	Q
Bit 29 - Bank requires common D-bank contingencies	H
Bit 28 - Allow this bank to process program bank contingencies	A
Bit 27 - Start bank on absolute 2048 word boundary	T

Preference value is as follows:

0 - Requires primary storage
1 - Requires extended storage
2 - Prefers primary storage
3 - Prefers extended storage
4 - No storage preference
5 - Requires Extended Mode

There is one BLT entry for each bank in the program. Entries are ordered by ascending BDI beginning with BDI 4.

For V-option banks (assign addresses but strip code), block count is 0. Block count is the number of 512-word blocks or 64-word blocks (if TYPE BLOCKSIZE64 was specified) in the bank.

BLT is immediately followed by the common bank list if it exists.

6.4. Common Bank List Format

Each word in the Common Bank List is formatted as shown in Table 6-4.

Table 6-4. Common Bank List Format

First BDI	Second BDI	Third BDI

This table contains the BDI for the EXEC bank descriptor table (BDT) of all common banks referenced within the program.

BDIs of common banks specified for initial basing will be included within this list.

T3 of Word 6 of the program file Absolute Element Table contains the number of third-word values contained in this list.

This table immediately follows the Bank Load Table.

6.5. Absolute Element Text and Access Control Word (ACW) Formats

The following section consists mainly of figures depicting the absolute element text and access control word (ACW) formats. An example of ACWs in an absolute element and segment load table entry formats is also included.

6.5.1. Absolute Element Text Formats

The absolute element text formats are shown in Figure 6-1 through Figure 6-4.

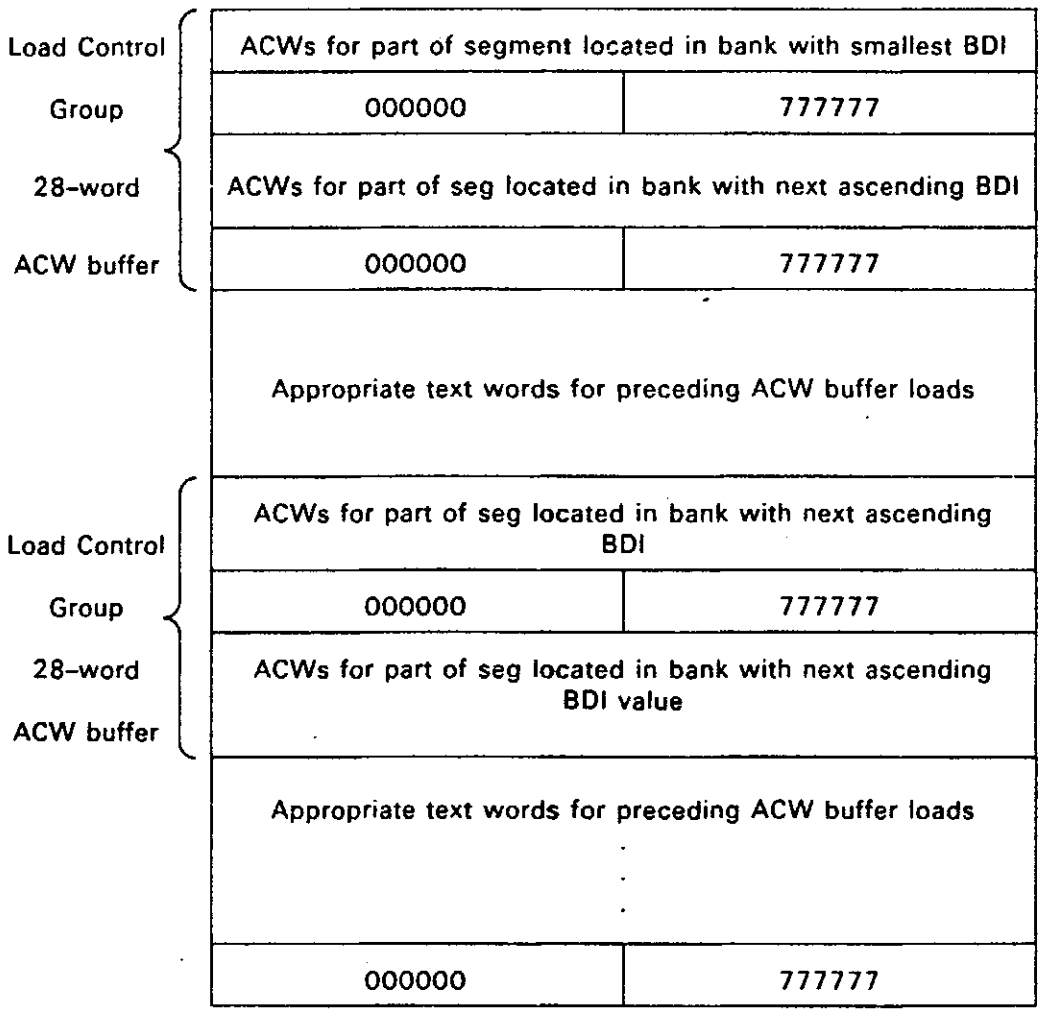


Figure 6-1. Example of Overlay Segments which Span Banks

NOTES: - *There is one EOL sentinel (000000 777777) for each bank which contains part of segment.*

- *SLT$ points to start; one SLT extension for each segment part (bank-named).*

- *ACWs and EOL sentinels may span sectors.*

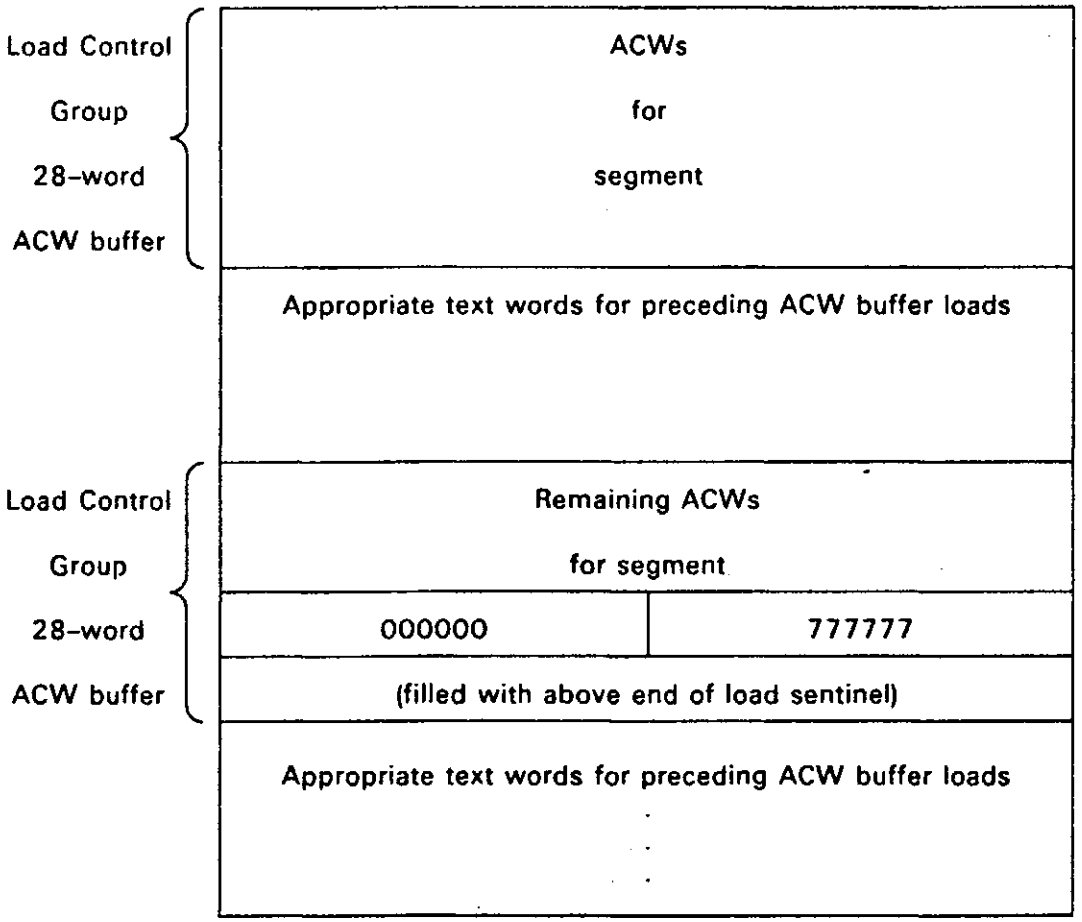


Figure 6-2. Example of Overlay Segments which Do Not Span Banks

NOTE: There is only one EOL sentinel.

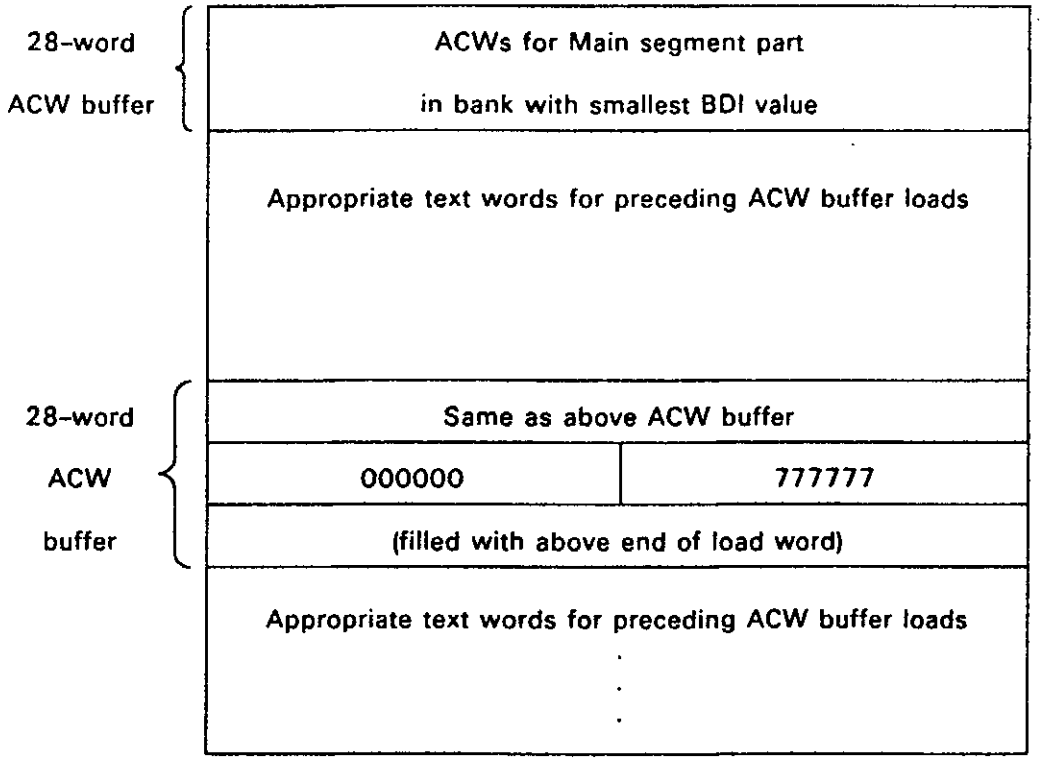


Figure 6-3. Main Segments in Non-initially Loaded Dynamic Banks

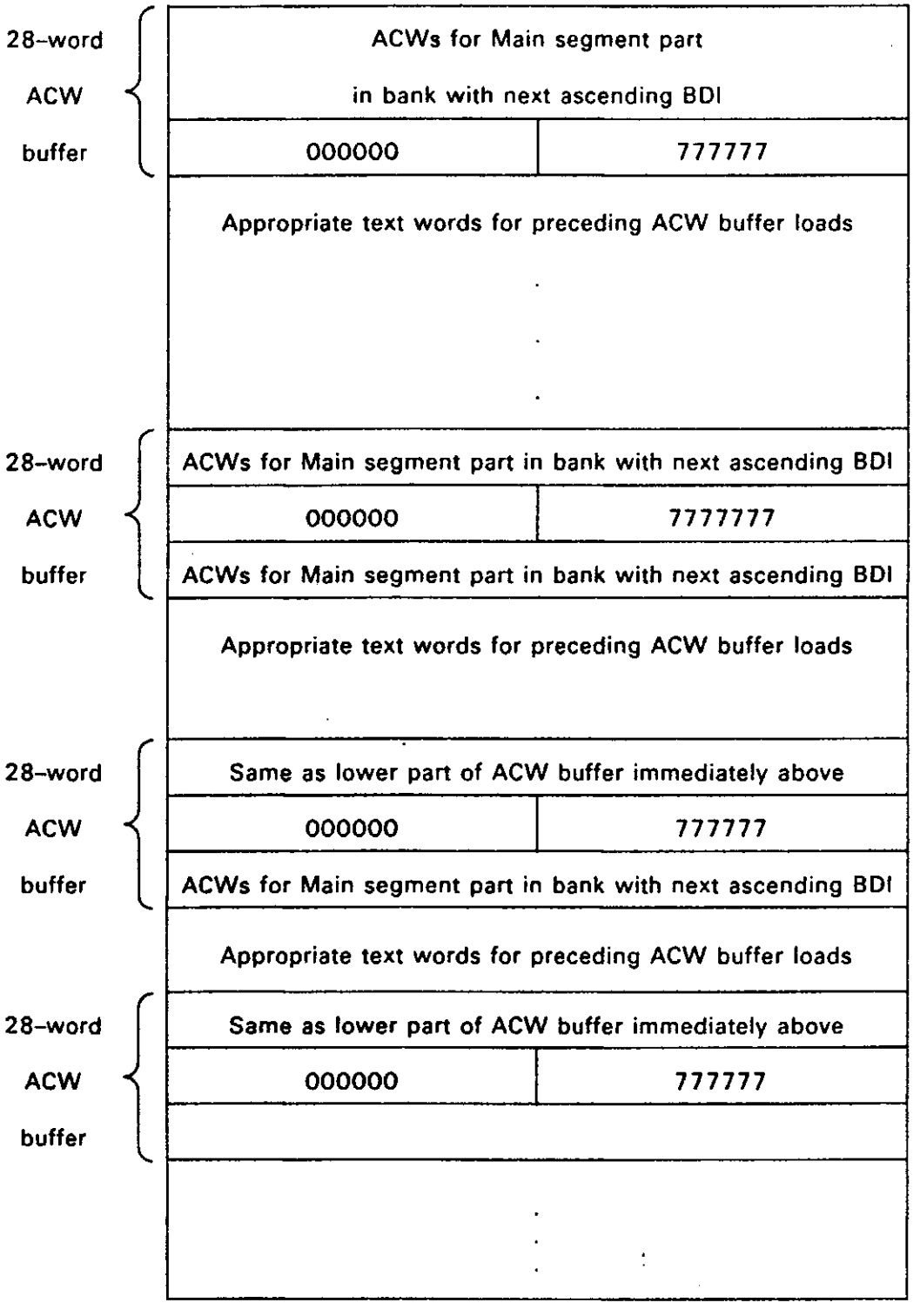


Figure 6-4. Main Segments in Initially Loaded Dynamic and Static Banks

The only case where ACWs for a segment spanning banks are not packed within any ACW buffer is when the ACWs are for the main segment of a non-initially loaded dynamic bank. That is, the first ACW for a non-initially loaded dynamic bank's main segment will always start on a sector boundary, as will its first text word.

6.5.2. ACW Formats

The normal ACW format is shown in Figure 6-5.

Figure 6-5. Normal ACW

number of words to load	program start address for load

The ACW in Figure 6-6 is used to zero fill areas of storage into which no text is initially loaded.

Each bank's main segment ACWs contain a zero fill ACW to fill storage with zeros from the end of the main segment to the end of the last storage block assigned to that bank.

Figure 6-6. Zero Fill ACW

1	0	number of words to zero fill	program start address for fill

The ACW in Figure 6-7 is an end-of-load sentinel. It marks the end of the set of normal and zero fill ACWs for each segment part within a bank.

Figure 6-7. End-of-Load Sentinel

000000	777777

The ACW in Figure 6-8 is for the relocation bits for a relocatable segment (RSEG) and immediately follows the end-of-load sentinel for the actual RSEG text ACWs.

Figure 6-8. RSEG Relocation Bits ACW

number of relocation words	000000

The ACW in Figure 6-9 is a NOP ACW. It is recognized and bypassed by the loader.

Figure 6-9. NOP ACW

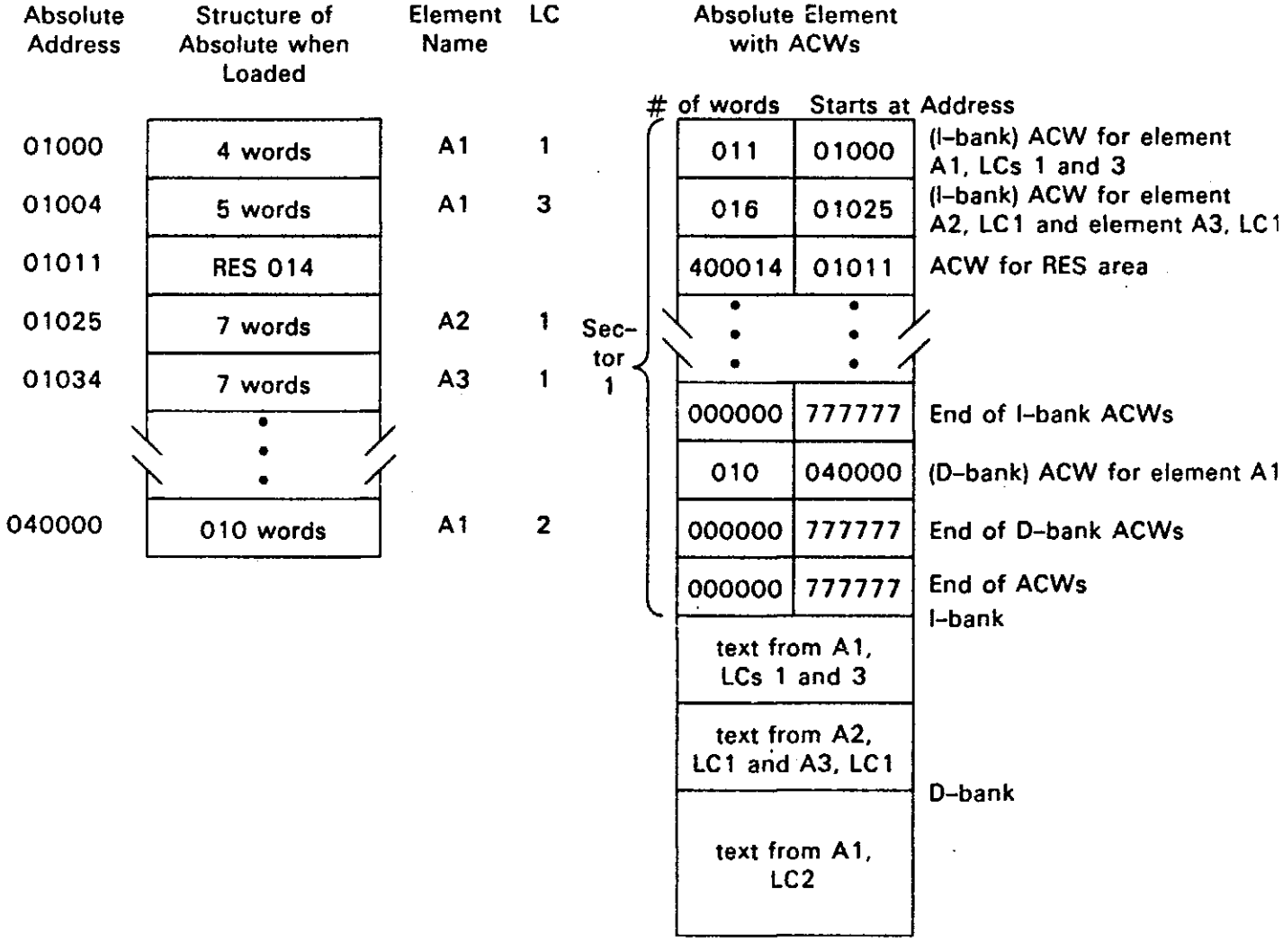
0	1		0

The ACW in Figure 6-10 is a NOP Common Block ACW. H2 contains the BDI of the bank into which common block text is to be loaded. The ACWs immediately following the NOP common block ACW specify the locations where the common block text is loaded within the indicated bank. An end-of-load sentinel will be found following the last ACW for the common block text load.

Figure 6-10. NOP Common Block ACW

6.5.3. Example of ACWs in an Absolute Element

Consider the hypothetical situation depicted in the following diagram.



Bit 35 of the ACW, when set, tells the loader to zero fill the RES area.

The sentinel 000000777777 shows:

1. The end of the ACWs for the I-bank, or D-bank within a sector,
2. The end of the ACWs if a sector is only partially filled (the rest of the sector would also be filled with the sentinel), and
3. If the last sector of ACWs is completely filled, a sector of sentinels follows it.

6.5.4. Segment Load Table Entry Formats

Two formats of the Segment Load Table (SLT) exist; one for bank-implied collections, and one for bank-named collections.

Table 6-5 shows the format for bank-implied collections.

Table 6-5. Segment Load Table Format - Bank-Implied

00	A	type	0	forward link to active segment
01	last I-bank address			first I-bank address
02	last D-bank address			first D-bank address
03	0		sector address of first load control group	

where:

A — Bit 35 set if segment is not loaded.

type

000		Main segment for bank-implied SLT format
010	("C")	Dynamic segment
027	("R")	Relocatable segment
024	("O")	Overlay segment

If type = 0 in the first SLT entry, the table only contains four-word entries as previously formatted.

Table 6-6 shows the format for bank-named collections.

Table 6-6. Segment Load Table Format - Bank-Named

00	A	type	0	forward link to active segment
01	last bank address			first bank address
02	BDI		0	extension index
03	0		sector address of first load control group	

where:

A	Bit 35 is set if segment is not loaded.		
type	022	("M")	Main segment for bank-named SLT format
	011	("D")	Dynamic segment
	027	("R")	Relocatable segment
	024	("O")	Overlay segment
BDI	BDI value for bank into which segment part is loaded.		

The forward link to the active segment is the address for the next Segment Load Table entry of a loaded segment. That is, all loaded segments are chained together by this link. All segments are chained off the main segment. Though the main segment is always first, the rest of the segments are chained in an arbitrary manner. The half words used for the forward links to the active segments are used by the EXEC.

The sector address of the first load control group points to the first group of ACWs. This also is used only by the EXEC.

Bank-named SLT formats may contain extension entries. An extension entry is necessary when a segment spans banks. When word 2 (H2) of the SLT is non-zero, it contains a link to the next SLT extension for the segment. The extension index is the program address of the word immediately preceding the first word of the extension entry.

Extension entry:

Index + 00	last bank address		first bank address
01	BDI	0	extension index

Each four-word entry and its extensions are linked in order of ascending BDI value.

The SLT four-word entry for the main segment never contains an extension index as no extension entries are built for the main segment.

For RSEG SLT entries, the format is the same in both the bank-implied and bank-named Segment Load Table.

RSEG entry:

00	A	027	0	forward link to active segment
01	last RSEG address			0
02	BDI		0	number of relocation words
03	0		sector address of first load control group	

6.6. Collector Generated Tables

The following tables exist in absolute elements and are accessible to the user via the Collector defined tags:

ENTRY$ - first address of the Entry Point Table
COMMN$ - first address of the Common Block Table
XREF$ - first address of the External Reference Table

If these tables do not exist, the values of these tags are zero.

These tables contain three word entries as described below. The COMMN$, ENTRY$, and XREF$ tables have a one word header of the format:

0	number of entries

6.6.1. ENTRY$ - Entry Point Table

The Entry Point Table (see Table 6-7) contains three-word entries describing only those entry points specified on the DEF directive.

Table 6-7. Entry Point Table Format

00	0	Number of entries
01	entry point name	
02		
03	value (absolute program address)	

If the entry point is in a segment designated for indirect loading, H1 of word 3 contains the address of the entry for the entry point in the Indirect Load Table and H2 of word 3 contains the absolute program address of the entry point.

6.6.2. XREF$ - External Reference Table

The External Reference Table (see Table 6-8) consists of three-word entries describing those references named on the REF directive. If any of these references is also an entry point, no entry is built in XREF$ for that reference.

Table 6-8. External Reference Table Format

00	0	Number of entries
01	external reference name	
02		
03	ER ERR$	

The external reference in the absolute text is assigned the address of the third word in XREF$. Therefore, if a jump has been made to the external reference, an ER ERR$ is done. This can be used for testing and debugging program modules.

6.6.3. COMMN$ - Common Block Table

When the REF or DEF directives are present, the Common Block Table is included in the absolute. The Common Block Table contains three-word entries as shown in Table 6-9.

Table 6-9. Common Block Table Format

00	0	Number of entries
01	common block name (BLANK $ COMMON for blank common)	
02		
03	length of common block	address of common block

6.6.4. SNAP Table

Each snapshot dump requested by a SNAP directive causes an entry to be made in the SNAP Table (see Table 6-10). This table is placed at entry point SNAP in element SNAP$. The instruction for the SNAP is replaced by SLJ SNAP$$ with the snap number in the a-field of the instruction.

Table 6-10. SNAP Table

00	SNAP$ control word	
01	frequency desired (from SNAP Data Image)	dump limit (from SNAP Data Image)
02	replaced instruction	
03	actual frequency counter	number of dumps taken

6.6.5. Indirect Load Table

Each reference made by a segment to an entry point in any I-banks of segments that are designated for indirect loading causes an entry in the Indirect Load Table (see Table 6-11).

Table 6-11. Indirect Load Table

00	SLJ	A
01	SLJ	B
02	actual entry point program address	index to segment load table (SLT$ entry defining this segment

Word 00:

A is one of the following entries:

IDJ$ Entry to indirect load routine for all jump commands except SLJ when zero fill is required on load and program does not contain DSEGs

IDJD$ Entry to indirect load routine for all jump commands except SLJ when zero fill is required on load and program does contain DSEGs

IDJA$ Entry to indirect load routine for all jump commands except SLJ when zero fill is not required on load and program does not contain DSEGs

IDJAD$ Entry to indirect load routine for all jump commands except SLJ when zero fill is not required on load and program does contain DSEGs

Word 01:

B is one of the following entries:

IDSLJ$ Entry to indirect load routine for SLJ commands when zero fill is required on load and program does not contain DSEGs

IDSJD$ Entry to indirect load routine for SLJ commands when zero fill is required on load and program does contain DSEGs

ISLJA$ Entry to indirect load routine for SLJ commands when zero fill is not required on load and program does not contain DSEGs

ISJAD$ Entry to indirect load routine for SLJ commands when zero fill is not required on load and program does contain DSEGs

6.7. Relocatable Segments

A relocatable segment allows the location of the RSEG within the program to be determined dynamically by the program during execution rather than at collection. Each relocatable segment is relocated relative to zero by the Collector. The instructions and data for all elements in an RSEG are collected together and assigned continuous addresses with odd location counters following even location counters.

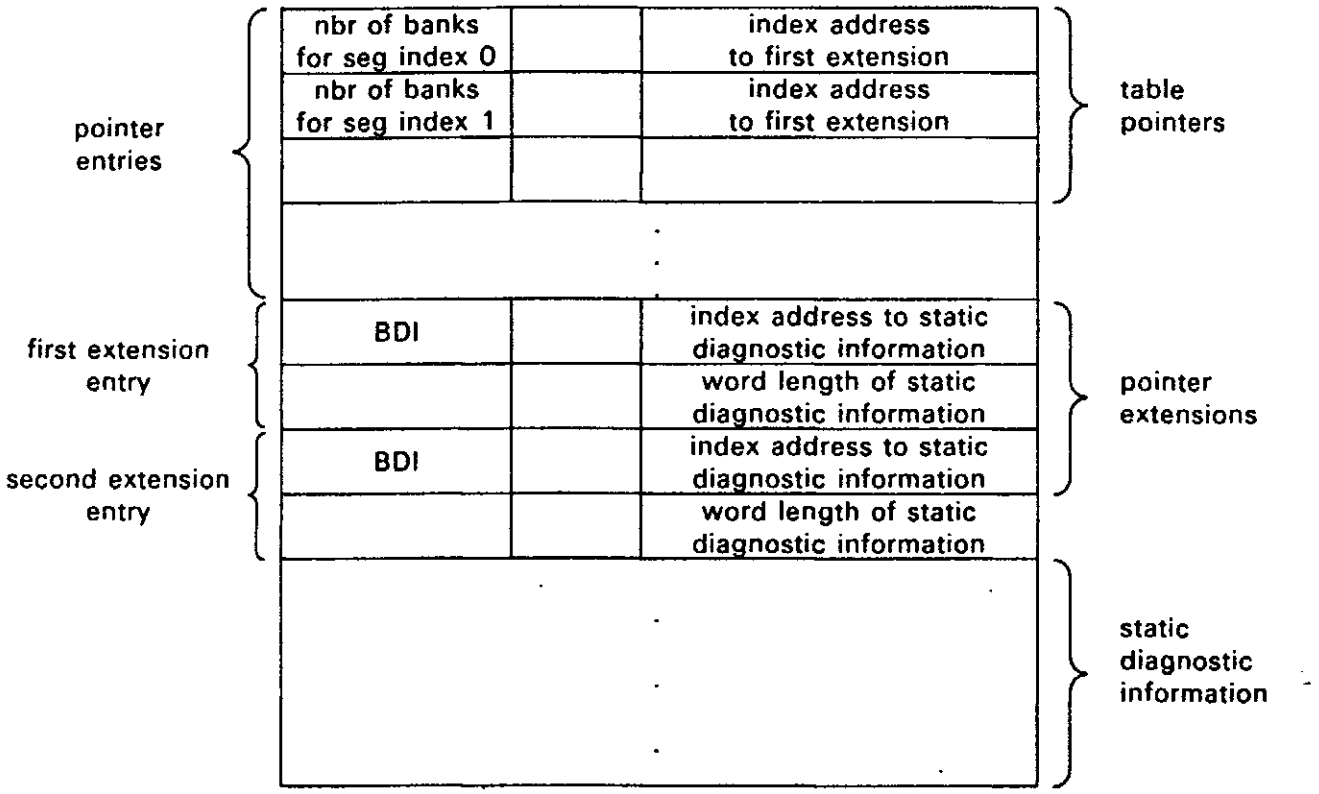
The relocation of an RSEG is done after the RSEG is loaded by adding the starting address of the RSEG to the right and/or left half of each word as marked by the Collector. The half words are not marked for load time relocation if the relocatable binary code specifies subtraction of a location counter or an undefined symbol that is not absolute.

The segment load table is marked to designate relocatable segments. A table is produced by the Collector that indicates what half words are to be relocated. Bit 35 of the first word of the table is 1 if the left half of Word 1 of the segment is to be relocated. Bit 34 is 1 to relocate the right half of Word 1 of the segment. Bit 33 is 1 to relocate the left half of Word 2 of the segment and bit 32 is 1 to relocate the right half of Word 2 of the segment. Therefore, each word in the relocation bit table represents 18 words in the relocatable segment.

6.8. Diagnostic Table Formats

The diagnostic table format is shown in Table 6-12.

Table 6-12. Static Diagnostic Pointer Table and Static Information



The pointer table is organized by ascending segment index (found in the Bank and Segment Index Value Table in the Collector listing). All index addresses are relative to the start of the table.

Number of banks for seg index *n* indicates the number of extension entries for this piece of diagnostic text. This number reflects the number of banks this segment spans.

The index address is relative to the start of the table for both extension and static diagnostic information.

The start of the pointer entries and the length of the pointer entries are found in the header table. If the pointer entries do not fill one sector, the extension entries begin on the next sector boundary.

6.8.1. Segment Name Table (SNT)

The Segment Name Table contains Segment Name Entries (see Table 6-13) and Extension Entries (see Table 6-14).

Table 6–13. Segment Name Table (SNT) Entry

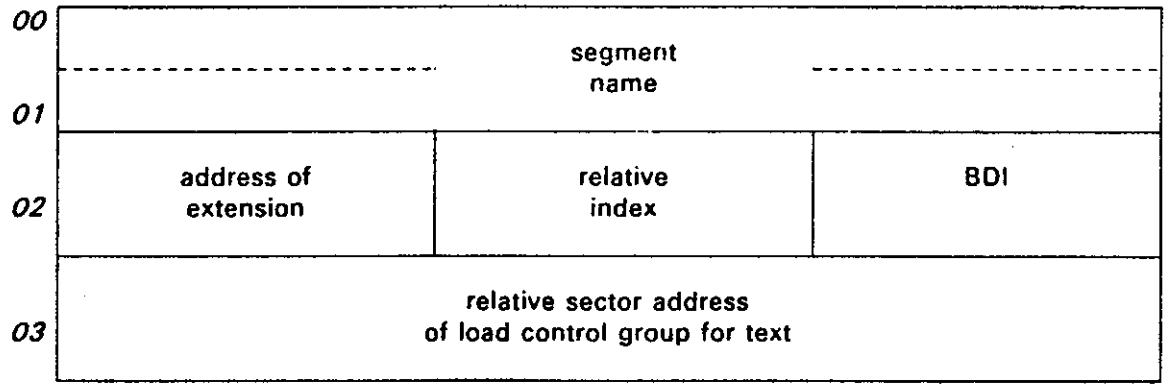
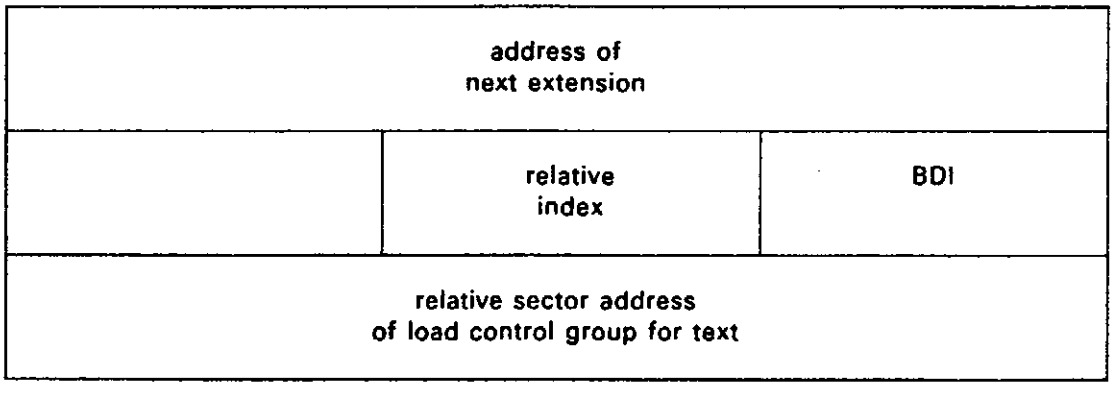


Table 6–14. Segment Name Table Extension Entries

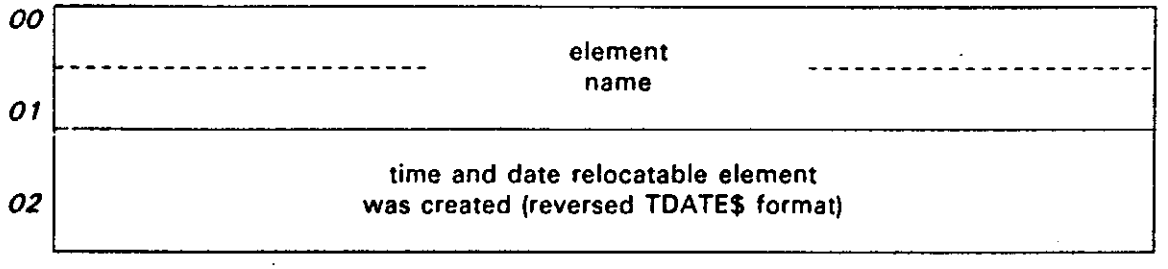


NOTE: Relative index points to the word in the load control group where this segment description begins.

6.8.2. Element Name Table (ENT)

The element name table format is shown in Table 6–15.

Table 6–15. Element Name Table (ENT)

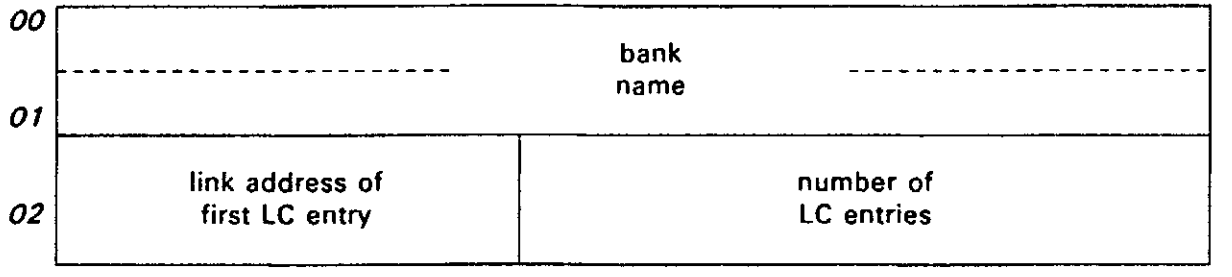


NOTE: The Element Name Table is created in alphabetical order.

6.8.3. Bank Name Table (BNT)

Each bank has an entry in the Bank Name Table (see 6-16). These entries are in ascending BDI order, and correspond one-to-one with the Bank Load Table entries.

Table 6-16. Bank Name Table (BNT)

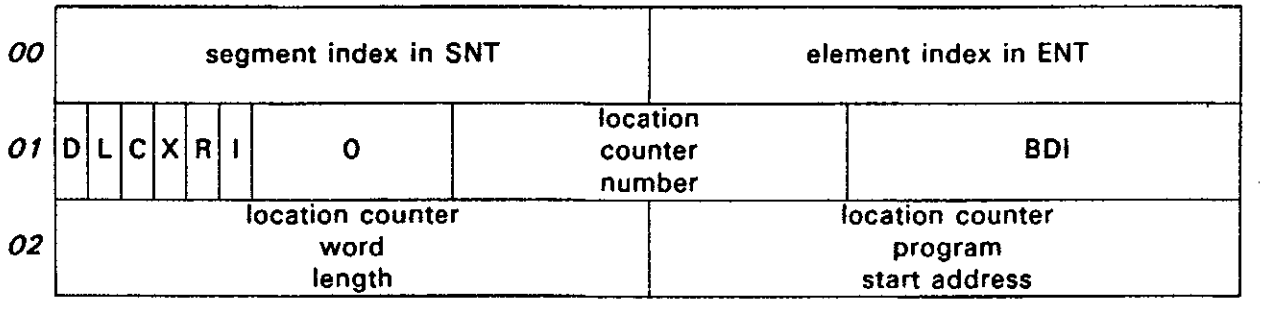


"Link address of first LC entry" is a word offset from the start of the LCT which describes the location counter.

6.8.4. Location Counter Table (LCT)

The location counter table is shown in Table 6-17.

Table 6-17. Location Counter Table Format (LCT)



Word 0 Segment index in SNT — The word offset from the beginning of the SNT of the segment containing this location counter.

Element index in ENT — The word offset from the beginning of the ENT of the element containing this location counter.

Word 1 D - Set if LC is in a D-bank
L - Set if LC is from an element in the file SYS$*RLIB$
C - Set if LC text is for a common block
X - Set if LC is the actual common block

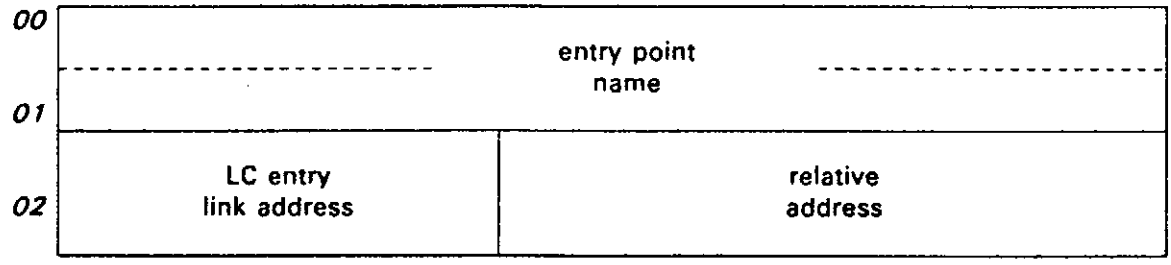
R - Set if LC text is in an RSEG
I - Set if LC contains INFO-8

If C is set, H2 of Word 2 contains an index to the LCT entry for the final LC assigned to the common block in the absolute element. If X is set, the LC number is 0. It is a logical conflict for both C and X to be set in the same entry.

6.8.5. Entry Point Name Table (EPNT)

The entry point name table is shown in Table 6-18.

Table 6-18. Entry Point Name Table (EPNT)

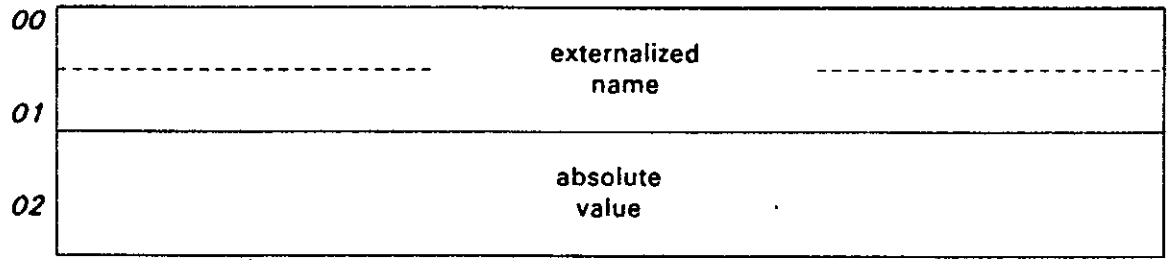


"LC entry link address" is the word offset from the start of the LCT which describes the location counter. Relative address is the address of the entry point relative to the start of the location counter. The Entry Point Name Table is created in alphabetical order.

6.8.6. Absolute Value Definition Table (ABSV)

The absolute value definition table is shown in Table 6-19.

Table 6-19. Absolute Value Definition Table (ABSV)



6.9. DIAG$ File - PMD Header Block Format

The postmortem dump (PMD) mode header block format in Table 6-20 is provided for the FLIT user. The format is internal to the Series 1100 Executive System and is subject to change without notice.

The header block is normally located at sector 1000. However, if diagnostic dumps have been used, then sector 0, word 5 contains the sector number.

Table 6-20. DIAG$ File - PMD Header Format

00	P	M	D	$	$	$
01	Internal Filename of Program File					
02	Containing the Absolute Element					
03	Program Header Table Address					
04	Type	Level	LPEB	ASA Address		
05	Reserved For Later Use					
06	Program Condition Word					
07		PCTBDI		Number of Banks Dumped		
010	BDI	Size		Address in DIAG$ File Relative to Header		
n	BDI	Size		Address in DIAG$ File Relative to Header		

Word	Description
00	This word contains a Fieldata sentinel for use in verifying that the table is present.
01-02	These two words contain the Fieldata internal filename of the program file containing the absolute element. These two words are initialized from the cell EU in the Program Control Table (PCT).
03	This word contains the relative sector address of the program header table (see Table 6-2) contained in the file indicated in Words 1 and 2. This word is initialized from the cell EX in the PCT.
04	This cell contains information relating to the last activity to terminate: S1 Program type. S2 Switching level. H2 PCT relative address of the last activity to terminate.

06 This word contains information pertaining to the condition of the program at the time the last activity terminated.

T1 Equal to zero implies that the program terminated normally.

S1 Equal to "E" implies that at least one activity of the program terminated in error.

S2 Equal to "E" implies that the last activity terminated in error. Equal to "X" implies that the last activity aborted.

07 The first bit is 1.

Q2 BDI of the PCT bank.

H2 This cell contains the number of banks, including the PCT, that were written to the DIAG$ file. Write-protected banks and dynamic banks not loaded at the time of termination were not dumped.

010-n The following cells describe the banks dumped.

Q1 BDI for this bank.

Q2 Final size of bank; the number of 512-word blocks. For single-bank programs collected with an implied collection, a zero length may appear for the missing I- or D-bank.

H2 This cell contains the sector address, relative to the PMD header block, for the start of this bank's area in the DIAG$ file.

7. Master File Directory (MFD)

For cataloged files, entries are constructed containing the identification and characteristics of each file. These entries are maintained by the system in the Master File Directory. The MFD consists of look-up table entries and directory items. A look-up table is linked to the cataloged files. Each filename and qualifier are folded to provide an index into the look-up table. The length of the look-up table is a system generation parameter. Directory items are 28-word areas used to store information needed to maintain a file's identity and description. Directory items are defined as:

- Search Item

  Used to locate pointers to lead items having the same index into the look-up table.

- Lead Item

  Used to provide the link between the qualifier/filename combination and each file of the F-cycle set. The lead item contains pointers to the main items.

- Main Item

  Used to store information that defines a file. There is one main item for each file of the F-cycle set. The main item contains a pointer to the granule item.

- Device Area Descriptor (DAD) Table/Reel Number Table

  A DAD table is comprised of entries that correlate the file relative location of the file's contents with the physical location of the file's contents. A reel number table contains a list of reel numbers assigned to a cataloged tape file.

A word in the look-up table contains one of the following types of entries:

- Zero

  A look-up table word is zero if no file set has an index equal to the word number.

- Pointer to Lead Item

  If only one file set has an index equal to the word number, the look-up table word has bit 35, the left-most bit, set to zero and points to the file set's lead item.

- Pointer to Search Item

  If more than one file set has an index equal to the word number, the look-up table word has bit 35, the left-most bit, set to one and points to the search item.

Figure 7-1 illustrates an MFD entry.

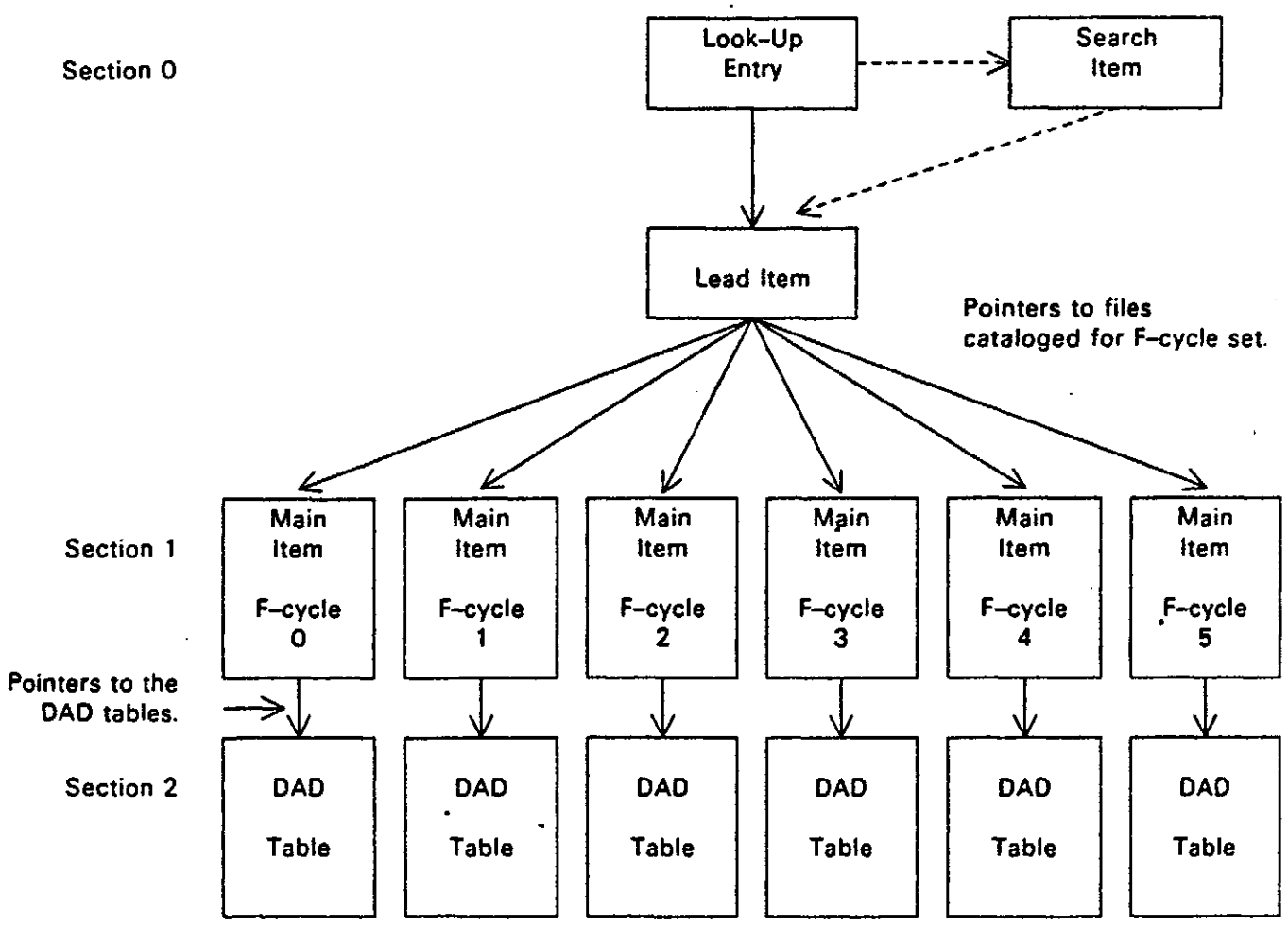


Figure 7-1. Example of an MFD Entry

7.1. Mass Storage File Addressing

A primary object of the MFD is to provide the user a convenient means to access a mass storage file. Different mass storage devices have unique addressing methods, a wide range of storage capacities and other significant differences at the hardware level. In addition, the Executive determines file placement according to current mass storage availability and a "best fit" algorithm, among other parameters. In order to unburden the user from needing to know where a file is physically placed and how to access that particular device, a file relative addressing mechanism is utilized.

The user accesses a file by referencing a filename and a location relative to the start of the file without regard to where the file text physically resides. The Executive determines the actual physical location of the file by using the Device Area Descriptor (DAD) tables. The primary unit of reference is the

three-word DAD entry, which describes a length of file space. File space described by a DAD can either represent allocated physical storage area on some device, or an expanse of unallocated file area which exists in the file's addressing structure, but for which no physical storage space has been allocated. This unallocated file space is commonly referred to as a "hole". Current Executive philosophy holds that unless otherwise directed, if a user writes into the first track of a file and the twelfth track, there is no need for the intervening ten tracks to be represented by allocated mass storage space. A DAD representing ten tracks of unallocated file space maintains the file relative addressing structure without utilizing costly mass storage space. It is also true that contiguous file relative mass storage space need not be physically contiguous. It is possible for tracks one, two and three of a file to be allocated on three different mass storage devices, and the user program can access these tracks with contiguous file relative addresses.

In order to accomplish this file relative to physical address transformation, it is necessary to be able to describe a point on mass storage with a concise notation, namely by using the Logical Device Address Table (LDAT) index and the device relative address.

7.2. Logical Device Address Table (LDAT)

LDAT is a resident Executive table that provides the logical ordering of all mass storage devices configured in the system. The Executive uses this ordering to address devices by a logical index.

The LDAT format is shown in Table 7-1.

Table 7-1. Logical Device Address Table (LDAT)

0	*unused*	*Length of LDAT*
1	*address of the unit status table associated with this LDAT index*	*address of the logical device unit status table associated with this LDAT index*
2		
n-1		
n		

7.3. Device Relative Address

A device relative address is a specific location on a device expressed in words. The first addressable word on the device is word 0 and all succeeding words follow in contiguous order.

7.4. Device Area Descriptor (DAD) Tables

The DAD tables are used by the Executive to correlate file relative addresses with physical mass storage addresses. Every mass storage file has at least one DAD table (see Table 7-2). On every I/O reference to a file, the Executive uses the file relative address as an index into a DAD table chain to find the physical location of the file relative address.

The primary unit of reference in a DAD table is the three-word DAD entry. A DAD is composed of four fields:

- Device Relative Address

  Defined in 7.3.

- Number of Words

  The number of words in a contiguous area.

- Flag bits

  Descriptors that define characteristics of the area.

- Device Index

  Specifies the device where the area resides.

Table 7–2. Device Area Descriptor (DAD)

Device Relative Address	
Number of Contiguous Words	
Flag Bits	Device Index

All addresses and lengths are specified in words. This removes equipment type dependencies and simplifies and shortens code that utilizes those fields.

7.4.1. Device Index

For fixed mass storage, the device index is an index into the Logical Device Address Table (LDAT). For removable disk mass storage, the device index is an index into the user pack-id table. A negative value in the device index field signifies an unallocated area in the file relative addressing structure. If a negative value does exist in the device index field, the device relative address is meaningless and may be non-zero.

The format of the user pack-id table is:

0	Length of User Pack-id Table	Number of Pack-ids
1	Number of Directory Sectors To Be Written To Pack	Index into LDAT
2		Index into LDAT
3		Index into LDAT
4		
n		

Thus, it can be seen that for removable disk, the device index points indirectly to the LDAT, via the user pack-id table. The user pack-id table resides in the Program Control Table (PCT) and contains one word for each pack assigned to the user.

7.4.2. Flag Bits

The flag bits in a Device Area Descriptor are:

Bit 18	Designates a dynamically detected bad spot.
Bit 19	Removable disk bit.
Bit 20	End of DAD table.
Bit 21	DNR (Do Not Read)
Bit 22	DNW (Do Not Write)
Bits 23-35	Reserved for future use.

NOTE: *Bits 21-22 are used only by TIP file control. For discussion refer to Series 1100 Transaction Processing, Programmer Reference, UP-8296, and Series 1100 Transaction Processing, Installation Guide, UP-8295* (applicable versions.)

7.4.3. DAD Tables

A DAD table contains all the information necessary to convert a file relative address to an absolute address. Two range words define the upper and lower limits of the file relative addresses described by the table. One of these range words gives the file relative address of the first word in the table while the other range word gives the file relative address of the last word described in the table (biased upward by one for ease of computation). Up to eight DAD entries per table define the physical location of the file space. Since each DAD entry contains the number of words described by the DAD, it is

a simple task to find any file relative address. First do a range check on a file relative word address to find the correct table, then add the number of words in each DAD to the starting range word of the table until the DAD entry containing the file relative address is found.

DAD tables also contain forward and backward directory links to logically contiguous DAD tables. In addition, when a DAD table is kept in main storage while a file is assigned, a forward link to the next DAD table in main storage is maintained, along with a flag that indicates whether the DAD table has been changed since it was last written out and the directory location of the DAD table. Table 7-3 shows the DAD table.

7.4.4. Mass Storage Allocation

Mass storage is allocated in granules of either 03400 words (track granularity) or 0340,000 words (position granularity).

Mass storage is allocated to a file as a result of an initial reserve request on an @ASG statement or as a result of a write into a previously unallocated granule of a file.

As a general rule, the Executive tries to allocate physically contiguous space for a logically contiguous file area. However, this is often impossible with a dynamically changing allocation scheme such as the Executive uses. This is especially true of files that have no initial reserve, where space is allocated dynamically as a user program writes into the file. A three-word DAD describes a mass storage area that can vary in size from one granule to an enormous number of granules. It is naturally more efficient for granules to be allocated contiguously, both in terms of directory space saved by using fewer DADs and in terms of the amount of computation needed to convert a file relative address to a physical address. For this reason, it is more efficient to use the initial reserve field on the @ASG or @CAT statement, especially where the size of the file is known and where most of the file would consist of logically contiguous data.

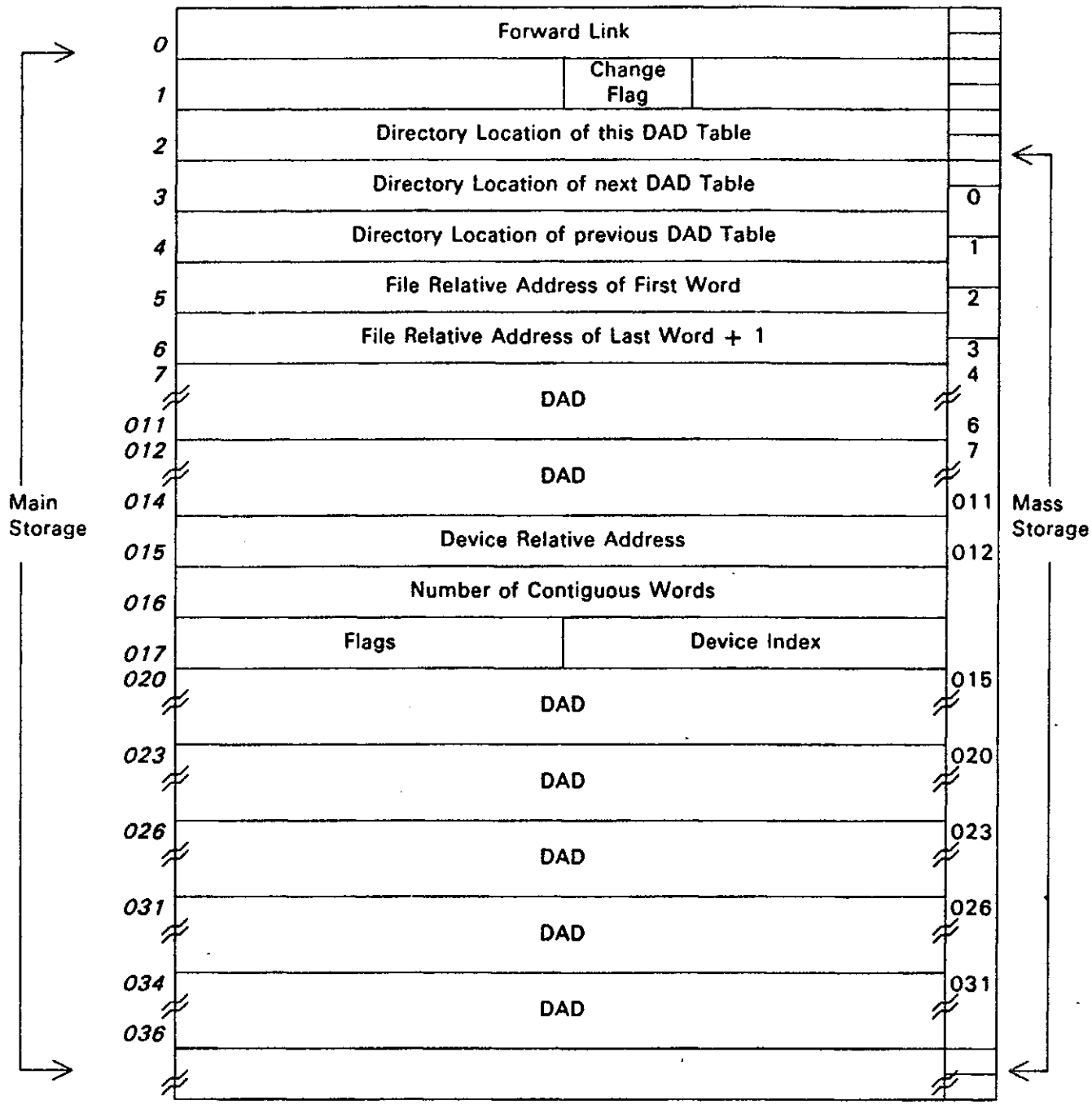
7.4.5. Master Bit Table (MBT)

The MBT (Table 7-4) is a bit map of mass storage space on a unit. Each bit represents one sector-formatted track. If a bit is set, the track is allocated. Each word represents 32 tracks. Two words represent one sector formatted position. Bit 35 represents the lowest numbered track, bit 4 is the highest numbered track in the word. The bottom four bits in each word do not represent any mass storage space. A checksum feature is available and when configured, the last word in the MBT contains the checksum value. The length of the MBT is variable and depends upon the size of the device. If the available space on a device does not end on a position boundary, the bits representing nonexistent space in the last position are set as allocated when the MBT is initialized.

If a high speed drum is used as the system drum, MBTs for all drum and sector-formatted type devices are placed on the system drum. Otherwise, they are located on the first directory track of the device.

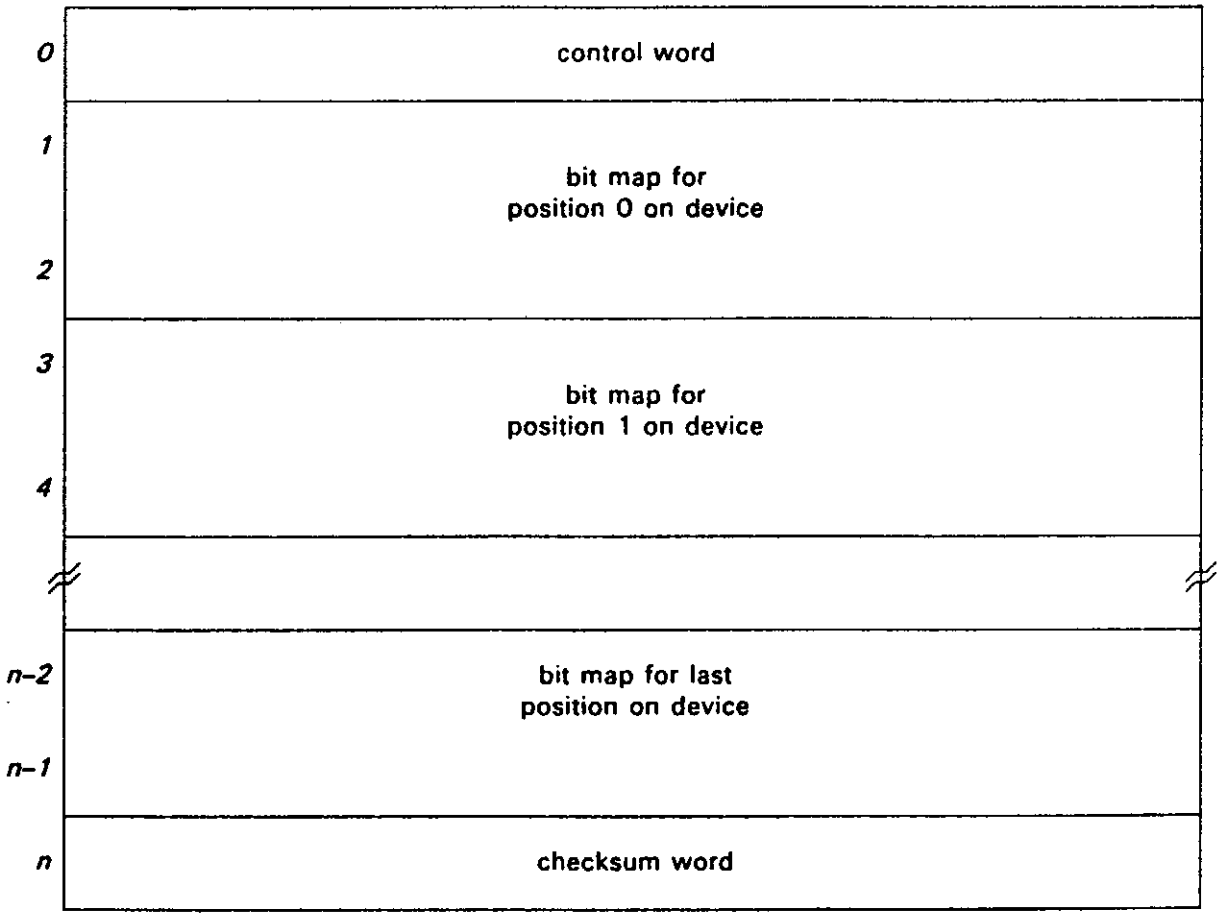
For disk type devices, MBTs are always kept on the first directory track. There are two MBTs for each disk type device. There is a hardware MBT and a software MBT. The hardware MBT is built when a disk is prepped. All defective tracks are marked as allocated in the hardware MBT. When a disk is initialized or recovered, the hardware MBT is used as a template for the software MBT so that defective tracks can never be allocated to user files.

Table 7-3. Device Area Descriptor Table



NOTE: Bit 33 of word 4 (in main storage) is New Format Flag (NFF).

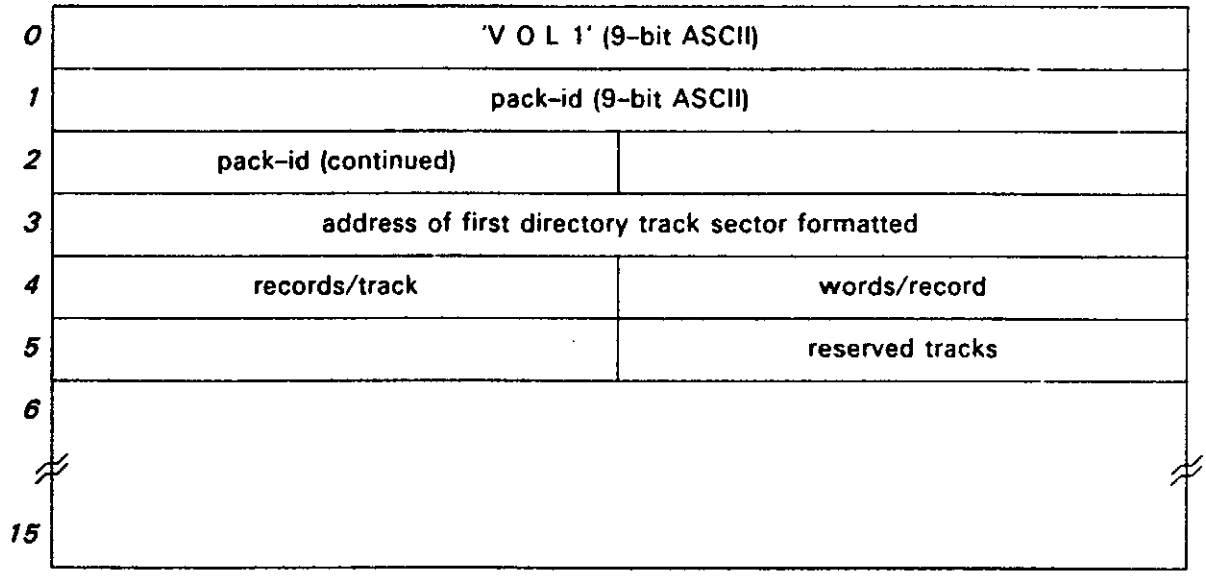
Table 7–4. Master Bit Table



7.4.6. Tables Unique to Disk Type Devices

Disk type devices have two tables utilized by the file recovery and pack verification segments of the Executive that do not exist on nonremovable storage such as high speed drums or sector–formatted drum devices. These tables, the VOL1 label block (Table 7–5) and sector 1 of the first directory track (Table 7–6), are used by the Executive to identify the disk pack and to locate key pieces of directory information. The tables are unique to disk type devices due to the unique problems presented by removable mass storage.

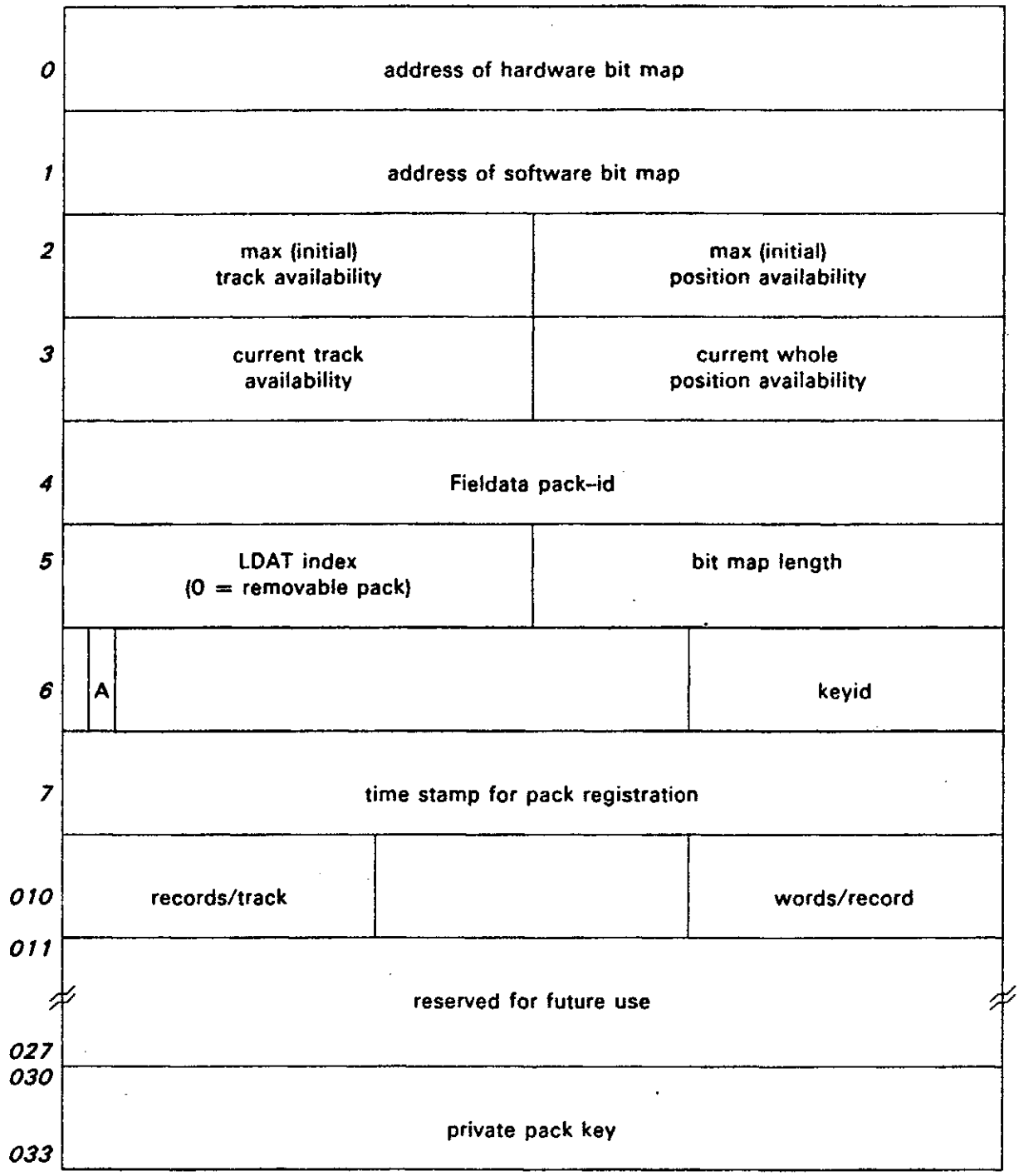
Table 7-5. Standard Disk Label Format



VOL1

This is the disk's label block. It contains the pack-id and prep factor of the pack. It also contains the location of the first directory track on the pack. VOL1 is in the same fixed location on every pack.

Table 7-6. Sector 1



Located in Sector 1 of the first directory track, this table contains the following information:

- Mass storage availability information
- Location of the hardware MBT and software MBT

- LDAT index of the pack if it is a fixed type pack. This LDAT index is assigned at the time the pack is initialized and stays unique to this pack through all subsequent recovery boots. If this field is zero, the pack is a removable disk pack.

- A – Word 6, bit 34: Activity flag used for removable packs. If set, it indicates that at least one file on the pack is currently assigned to a user run.

- A time stamp showing the last time a removable pack was registered. This is utilized by the automatic pack verification routine.

7.4.7. Mass Storage ID Table

The Mass Storage ID (MSID) table exists within the Master Configuration Table (MCT) and is used on recovery boots to determine if uninitialized mass storage devices are being added to the system. The MSID table has a one word entry for every mass storage device in the system. This one-word entry is indexed to by the LDAT index of the device. The word contains the logical device name of the device, or, for fixed disk units, contains the pack-id of the fixed pack carrying that LDAT index. The MSID table is used on recovery boots to guarantee that the indexing structure is rebuilt accurately.

On tape recovery boots, the MCT is not recovered and the MSID table is lost. Therefore serious mass storage problems can result from modifying the mass storage configuration while doing a tape recovery boot. When doing a tape recovery boot, be sure that no mass storage devices are added to the configuration either through a change in system generation (SYSGEN) parameters or by using UP keyins to include previously downed units.

7.5. MFD Structure

The MFD consists of many linked directory sectors. These sectors are maintained on mass storage within track size buffers referred to as directory tracks. A file's directory information is maintained on the same device as the file's data. This is done to safeguard against total MFD loss upon a device failure. Hence, at least one directory track exists for every mass storage device.

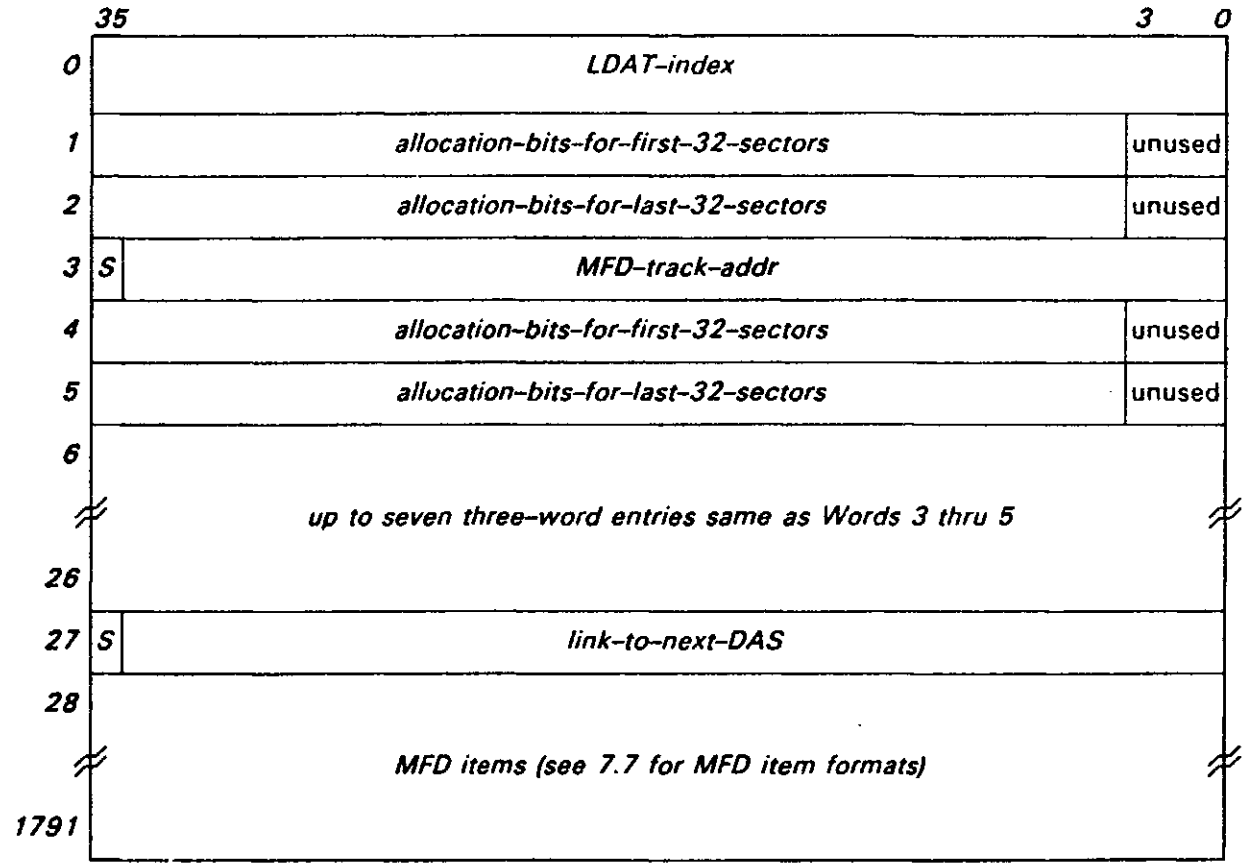
7.5.1. Directory Allocation Sector

A directory track is allocated in sector increments as required. When all 64 sectors of a directory track are allocated another track is acquired. The list of the acquired directory tracks and a map of allocated directory sectors within each directory track is maintained within a mass storage sector called the Directory Allocation Sector (DAS).

A DAS will hold information for up to nine directory tracks. One word is used to hold the track address and two words are used for a 64-sector bit map for each directory track. The last word is used as a forward link address to the next DAS for the device if required. Hence, every ninth directory track on a device starting with track zero has a DAS in sector 0.

The format of a DAS within a directory track is:

Table 7-7. Directory Allocation Sector (DAS)



NOTE: Throughout this section the word "unused" implies that the field so designated is not guaranteed to contain or maintain a specific value. Such fields may be assigned in the future, however.

Word 0

Bit 33 used for New Format Flag (NFF).

Word 1

Allocation bits for the first 32 sectors of this track. Bit 35 corresponds to sector 0, bit 34 to sector 1, and so on. The bit is set if the corresponding sector is allocated.

Word 2

Allocation bits for the last 32 sectors of this track. Bit 35 corresponds to sector 32, bit 34 to sector 33, and so on. The bit is set if the corresponding sector is allocated.

Word 3

Address of the next MFD track. This track does not contain a DAS in sector 0. If this word is negative (S = 1), then an MFD track does not exist for this entry.

Word 4

Same as word 1.

Word 5

Same as word 2

Word 27

Address of the next MFD track with a DAS in sector 0. If this is negative, there are no more MFD tracks for this unit.

7.5.2. Removable Disk

The directory for each removable disk pack is a separate entity. The main difference between a removable disk directory and the MFD is that the removable disk directory has no look up table, no search items and no lead items. These tables are not required because file assignment is not accomplished through direct use of the pack directory.

An Executive automatic pack registration routine registers the files contained on a removable pack when the pack is mounted. The Executive uses the directory information on the pack to create directory items for all the files on the pack in the fixed storage directory. After a pack is registered, the fixed storage directory is used to control any references by user runs to files on the pack.

7.6. General Description of the MFD File

The directory file is a temporary, sector-addressable file assigned to the Executive. The file spans all mass storage units, with at least one track allocated on each unit. The logical addresses of the file are partitioned into ranges of octal 10,000 tracks each, with a range of addresses dedicated to each mass storage device in the system. The file relative addresses of all directory tracks on a unit fall within this range. Therefore, it is possible to look at a file relative directory item address and determine which physical unit the address resides on; and units can be added or removed without upsetting the logical addressing structure of the file.

7.6.1. DAS-Relative Addresses

A DAS-relative address is equal to the depth of a directory track into a DAS table chain. Thus, the first directory track on a device is DAS-relative track 0 on the device, the track contained in the fourth track address cell in a DAS is DAS-relative track 3, and so on.

7.6.2. File Relative Address

Directory item addresses are file relative addresses. A directory address is derived from the LDAT index of the unit being allocated on, the DAS-relative location of the track being allocated on, and the sector number within the track.

A file relative directory link address format is:

S1	S2 & S3	S4 & S5	S6
descriptor bits	unit-separator (LDAT index)	DAS-relative-directory-track	sector number

where:

S1 is used for descriptor bits, which are described in 7.7.

Bits 29 - 18 (S2 & S3) indicate which unit the file relative address is on. This portion of the file relative address is called the Unit Separator, and the value in this field corresponds to the LDAT index of the device.

Bits 17 - 6 (S4 & S5) describe the DAS-relative directory track within the addressing range of octal 10,000 tracks dedicated to the unit.

Bits 5 - 0 (S6) describe the sector number within the track.

Thus, all the sector addresses from octal 1,000,000 to 1,777,777 describe unit 1; the addresses from octal 2,000,000 to 2,777,777 describe unit 2; and so on.

The addressing range of the directory file is then:

4096 directory tracks per unit. (A maximum of 261,688 directory items per unit.)

4095 mass storage units per system.

7.6.3. Directory Link Addresses on Removable Disk

The portion of the directory item addresses containing the LDAT index is zero for removable disk directory items. The LDAT index of the unit being referenced is added to H1 of the directory address before an I/O request is initiated. The correct LDAT index is retrieved from the user pack-id table.

It is useful to keep in mind that the directory file is not a removable disk file. The directory space on a removable disk is dynamically included in the directory file as fixed storage space while the removable disk is online.

7.7. Directory Item Formats

The MFD contains directory items for each cataloged file in the system. The content and format of the MFD are subject to change without prior notice. The directory item description is as follows:

Bits 35 through 32 in Word 0 of each directory item have the following significance:

Bit	Description
35	When set to 0, indicates that Word 0 of this sector contains a link address to the next sector.
34–32	001_2 - Search item
	010_2 - Lead item
	100_2 - Main item
	000_2 - DAD tables, or sectors 1–n of main item, or sector 1 of lead item. (See Tables 7–8 through 7–17 for examples of these items.)

Table 7-8. Search Item

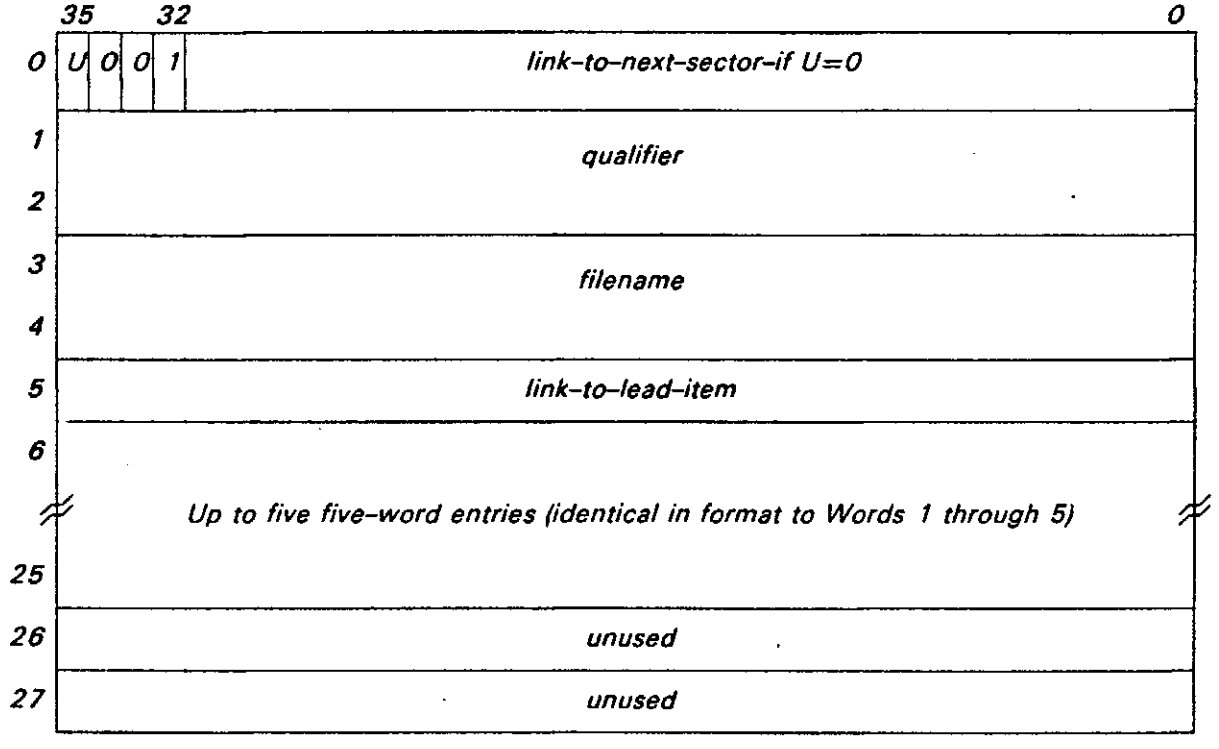
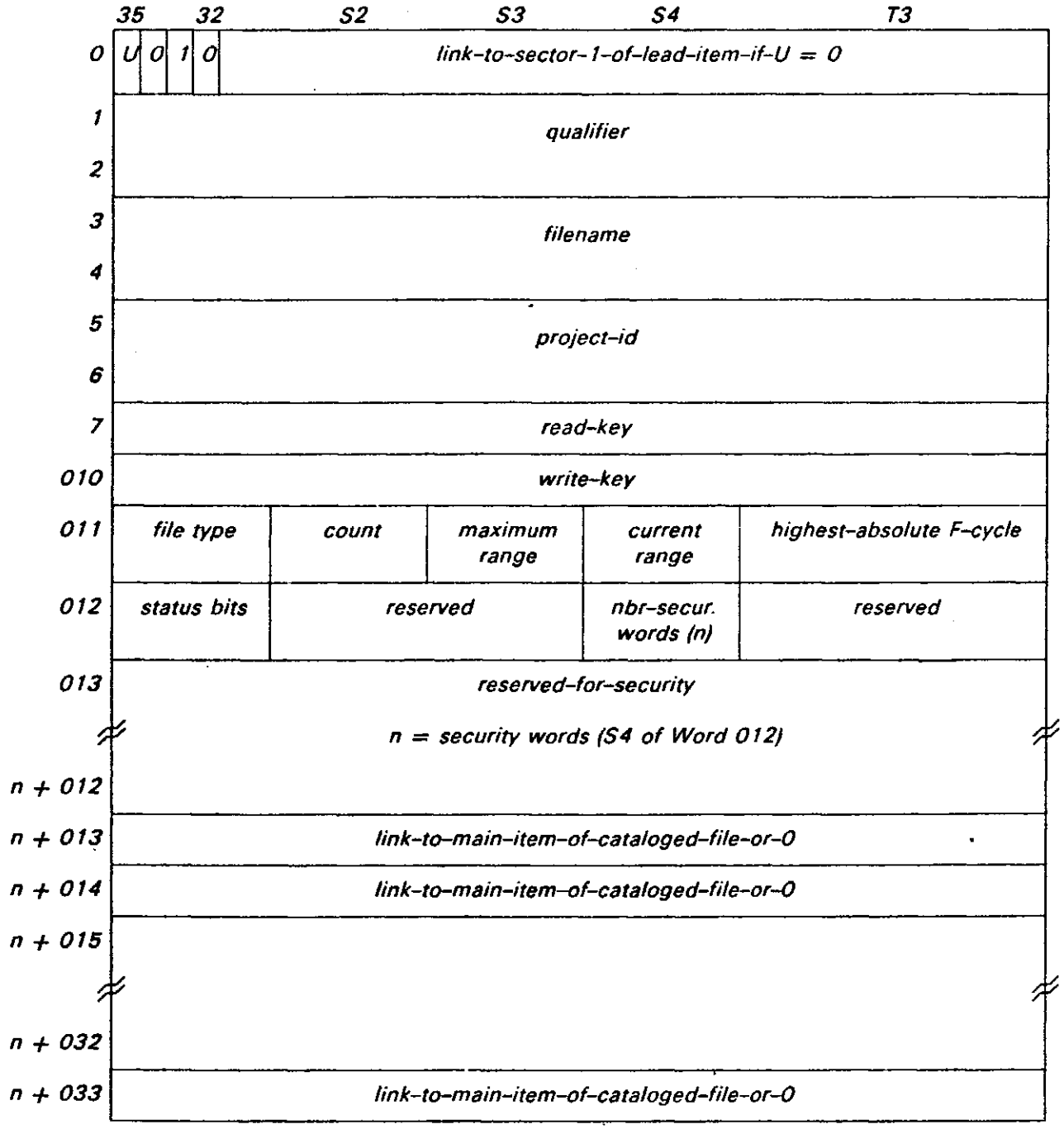


Table 7-9. Lead Item - Sector 0



Word 9

file type Type of device that F-cycles describe:

000	Mass storage
001	Magnetic tape
040	Removable disk

count Current number of F-cycles in the lead item.

maximum-range Maximum number of F-cycles permitted for the file.

current-range Range of absolute F-cycles currently in this lead item; (highest -absolute F-cycle)—(lowest-absolute F-cycle) + 1.

highest-absolute F-cycle The absolute F-cycle number whose link is in word (11 + number of security words) or would be there if it existed.

Word 10

status-bits These bits, when set, indicate:

35	@ASG,G	Guarded file
34	A (+1) F-cycle currently exists (link in word (11 + number of security words))	
33-30	Reserved	
29	Absolute maximum F-cycle	
28-24	Maximum F-cycles in lead item sector 0	
24	First link in lead item extension	
24-27	Maximum F-cycles in lead item sector 1	

nbr-secur words Number of security words, this number is used as the relative start of main item links in the lead item.

Word (11 + Number of Security Words)

Link to the main item that describes the highest-absolute F-cycle. If this absolute cycle does not exist, the entry is 0.

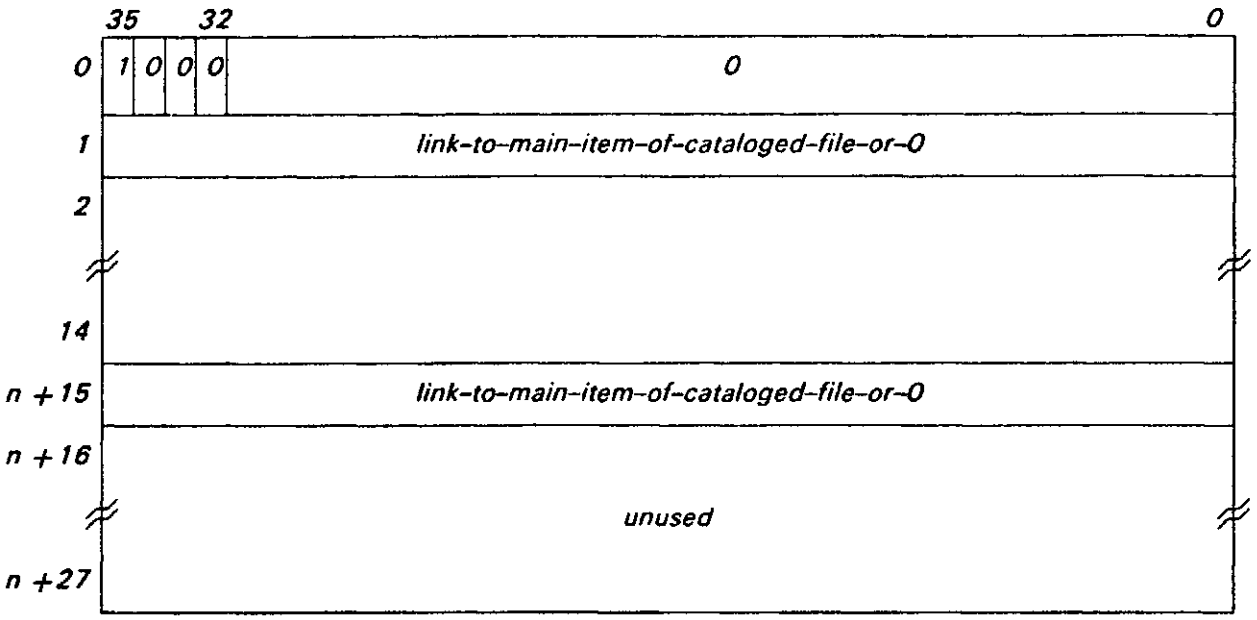
Word (12 + Number of Security Words)

Link to the main item that describes a file whose absolute F-cycle number that is one less than the highest-absolute-F-cycle. If this absolute cycle does not exist, the entry is 0.

Word 27

Link to the main item that describes a file whose absolute F-cycle number is (16 minus the number of security words) less than the highest-absolute-F-cycle. If this absolute F-cycle does not exist, the entry is 0.

Table 7-10. Lead Item - Sector 1



Word 1

Link to the main item that describes a file whose absolute F-cycle number is (17 minus the number of security words) less than the highest-absolute-F-cycle. If this absolute F-cycle does not exist, the entry is 0.

Word (15 + n, n = Number of Security Words)

Link to the main item that describes a file whose absolute F-cycle number is 31 less than the highest-absolute-F-cycle. If this absolute F-cycle does not exist, the entry is 0.

Table 7-11. Mass Storage File Main Item - Sector 0

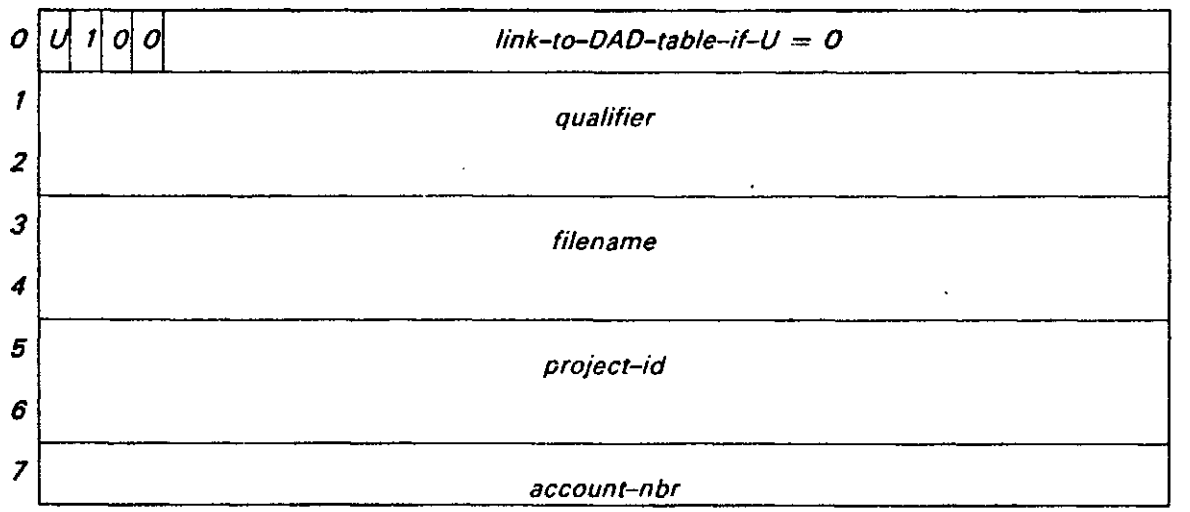


Table 7-11. Mass Storage File Main Item - Sector 0 (continued)

8				
9	*reserved*			
10	*time-of-unload-or-first-write-following-backup-item (TDATE$ format)*			
11	*disable flags*	*link-to-lead-item*		
12	*descriptor-flags*	*reserved*	*reserved for site use*	
13	*PCHAR flags*	*link-to-sector-1-of-main-item*		
14	*assign mnemonic*			
15	*symbiont-link-to-initial-SMOQUE-entry*		*total-nbr-of-times-this file-has-been-assigned*	
16	*run-id-or-EXPOOL-addr*			
17	*reserved*	*inhibit flags*	*nbr-of-runs-currently assigned-to-this-F-cycle*	*absolute-F-cycle-nbr*
18	*date-and-time-current-assignment-started-or last-assignment-ended (TDATE$ format)*			
19	*date-and-time-of-cataloging (TDATE$ format)*			
20	*initial-nbr-of-granules reserved-for-this-assignment*		*nbr-of-granules-of-quota-group-1*	
21	*maximum-nbr-of-granules*		*nbr-of-granules-of-quota-group-2*	
22	*highest-granule-nbr-assigned*		*nbr-of-granules-of-quota-group-3*	
23	*highest-track-written*		*nbr-of-granules-of-quota-group-4*	
24	*unused*		*nbr-of-granules-of-quota-group-5*	
25	*unused*		*nbr-of-granules-of-quota-group-6*	
26	*unused*		*nbr-of-granules-of-quota-group-7*	
27	*user-unit-selection-indicators*		*nbr-of-granules-of-quota-group-8*	

Word 10

time-of-first-write following-unload or-backup-time

If word 12, bit 35 = 1, then word 10 contains the unload time. If word 12, bit 35 = 0, then word 10 contains the time of the first write after the backup file was created or all zeros.

Word 11

disable-flags Bits 35 through 32, when nonzero, have the following significance:

1100_2	Facilities reject (file disabled because it was destroyed)
1010_2	Facilities warning (file disabled because of an incomplete write)
1001_2	SECURE processor reject (file disabled because it was rejected by the SECURE processor)

Word 12 *NOTE: Bit 35 is always set for a disabled file. The other bits may be ORed if more than one condition applies.*

descriptor-flags These bits, when set, indicate:

Bit	
35	Unloaded
34	Backed up
33	Reserved
32	Has lapse entries
31	Bad main item sector 1
30	Old item format (created on Executive Level 32 or earlier)
29	Tape file
28	Communication between recovery and MFD maintenance routines for removable disk files. Extension sectors of the main item may have been changed or created on permanent storage. This assures the subsequent update of extension sectors on the disk packs.
27	Removable disk file
26	File is to be made write-only
25	File is to be made read-only
24	File is to be dropped

Word 13

PCHAR-flags These bits, when set, indicate:

Bit	
35	Position granularity
34	Not used
33	Word-addressable mass storage
32	Prefers high-speed mass storage

Word 14 The assign mnemonic used on the @ASG or @CAT statement to create the file.

Word 17

inhibit-flags These bits, when set, indicate inhibit options on @ASG control statements as follows:

Bit		
29	@ASG,G	Guarded file
28	@ASG,V	Inhibit unload
27	Not @ASG,P	Private file
26	@ASG,X	Exclusive use
25	@ASG,W	Write-only
24	@ASG,R	Read-only

Word 27

user-unit selection-indicators These bits, when set, indicate:

Bit	
35	file created via device placement
34	file created via control unit placement
33	file created with logical placement (vs. absolute)
32	file is spread across devices
30–18	LDAT of initially selected device

Table 7-12. Mass Storage File Main Item – Sector 1

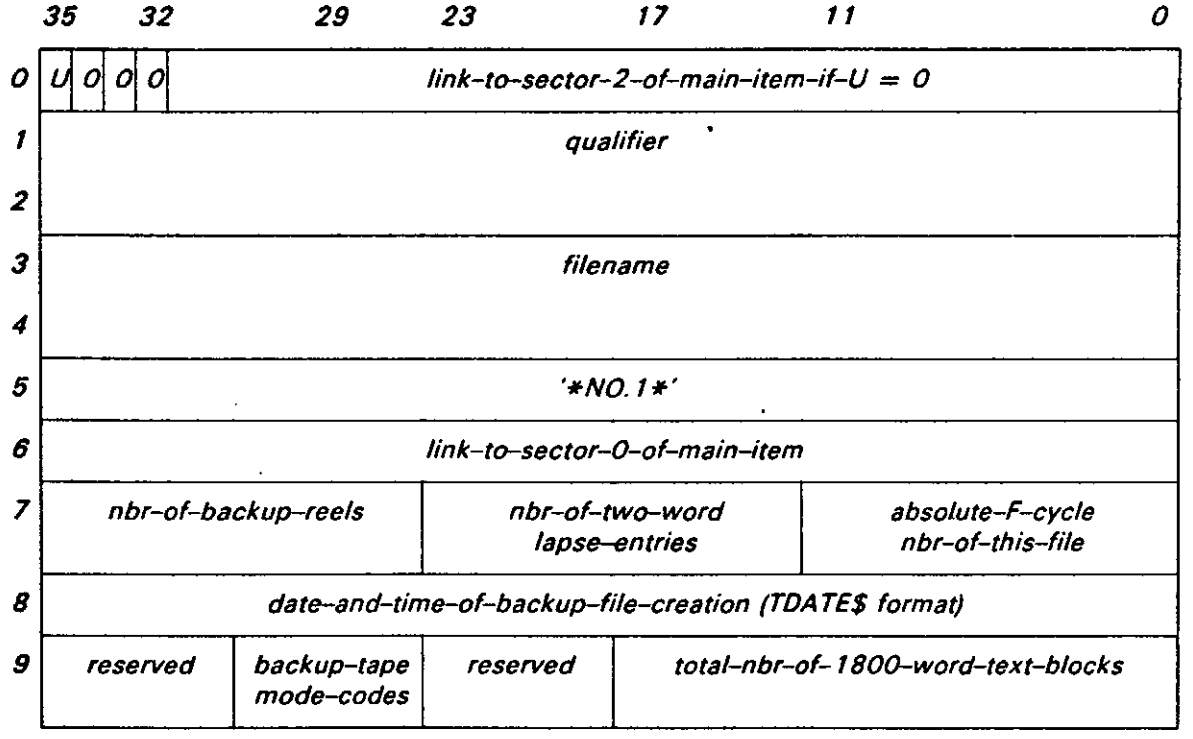


Table 7-12. Mass Storage File Main Item - Sector 1 (continued)

10	*tape-noise-constant*	*starting-file-position of-first-backup-reel*	*nbr-of-words written-in-last-block*
11	*reel-nbr-of-first-reel-of-backup-file*		
12	*reel-nbr-of-second-reel-of-backup-file*		
13 14	*first-lapse-entry-or-all-zeros*		
15 16	*second-lapse-entry-or-all-zeros*		
17	*reserved*		*nbr-disk-pack-entries*
18 19	*first-disk-pack-entry*		
20 21	*second-disk-pack-entry*		
22 23	*third-disk-pack-entry*		
24 25	*fourth-disk-pack-entry*		
26 27	*fifth-disk-pack-entry*		

Word 9

backup-tape-mode-codes Indicates type of peripheral containing file. Same as equipment codes. These octal bits when set indicate:

002	Software translate
004	Hardware translate
010	Low density
020	Medium density
030	High density
040	Even parity

Words 13–14

first-lapse-entry or-all-zeros

Word 13: Time of first write following backup file creation
Word 14: Time of REVERT when backup copy becomes primary copy
Date and time is given in TDATE$ format.

Word 18 – 19

first-disk-pack-entry

Word 18: Pack-id of the disk pack (six characters in Fieldata)
Word 19: Link to main item on disk pack

Table 7-13. Main Item – Sectors 2-n

0	U	0	0	0	*link-to-next-sector-of-main-item-if-U = 0*	
1	*qualifier*					
2						
3	*filename*					
4						
5	*'*NO.N*'*					
6	*link-to-previous-main-item-sector*					
7	P	B	L	*reserved*		*absolute-F-cycle-nbr*
8	*If P bit is set, up to 10 disk pack entries*					
≈	*If B bit is set, up to 20 backup file reel numbers*					≈
27	*If L bit is set, up to 10 lapse entries*					

*NOTE: The characters "*NO.N*" in Word 5 are exactly that for all sectors 2–n. n does not equal N.*

Table 7-14. Tape File Main Item – Sector 0

	35			32	
0	U	1	0	0	*link-to-reel-table-if-U = 0*
1	*qualifier*				
2					
3	*filename*				

Table 7-14. Tape File Main Item – Sector 0 (continued)

4				
5 6	*project-id*			
7 8	*account-nbr*			
9 10	*unused*			
11	*disable flags*	*link-to-lead-item*		
12	*descriptor-flags*	*unused*	*reserved for site use*	
13	*link-to-sector-1-of-main-item*			
14	*assign mnemonic*			
15	*unused*		*total-nbr-of-times-this file-has-been-assigned*	
16	*run-id-or-Expool-addr*			
17	*unused*	*inhibit-flags*	*nbr-of-runs-currently assigned-to-this-F-cycle*	*absolute-F-cycle*
18	*date-and-time-current-assignment-started -or-last-assignment-ended (TDATE$ format)*			
19	*date-and-time-of-cataloging (TDATE$ format)*			
20	*density*	*format*		*nbr-of-reels-cataloged*
21		*cataloging options*		noise-constant
22 23	*processor/tape-translator-mnemonics*			
24 25	*unused*			
26	*reel-nbr-of-first-reel-cataloged*			

Table 7-14. Tape File Main Item – Sector 0 (continued)

27	*reel-nbr-of-second-reel-cataloged*

Word 11

disable-flags bits 35–32

1100_2 Facilities reject (file disabled because it has been destroyed)

1010_2 Facilities warning (file disabled because of an incomplete write)

1001_2 SECURE processor reject (file disabled because it was unable to be loaded by the SECURE processor).

Word 12

descriptor-flags These bits, when set, indicate:

Bit	
35	Unused
34	Backed up
33	Backup file cannot be read
32	Unused
31	Bad main item sector 1
30	Old item format (created on EXEC Level 32 or earlier)
28–29	Tape file
27	Unused
26	File is to be made write-only
25	File is to be made read-only
24	File is to be dropped

Word 13

link-to-sector-1 of-main-item Sector 1 of main item for tape files is identical in format to that for mass storage files except there are no disk pack entries.

Word 17

inhibit-flags These bits, when set, indicate inhibit options on @ASG control statements as follows:

Bit		
29	ASG,G	Guarded file
28	@ASG,V	Inhibit unload
27	Not @ASG,P	Private file
26	Unused	
25	@ASG,W	Write only
24	@ASG,R	Read only

Word 20

density For seven track tapes, the possible values and their meanings are:

01	200 FPI
02	556 FPI
03	800 FPI

For nine track tapes, the possible values and their meanings are:

01	800 FPI
02	1600 FPI
03	6250 FPI

format The possible values and their meanings are:

001	Data converter
002	Quarter word
004	Six-bit packed
010	Eight-bit packed
020	Even parity
040	Set to nine track tape

Table 7-15. Removable Disk Pack Main Item (Sector 0)

	35	32	S2	S3	S4	T3
0	U	1	0	0	link-to-DAD-table-if-U = 0	
1	qualifier					
2						
3	filename					
4						
5	project-id					
6						
7	account-nbr					
8						
9	unused					
10	time-of-unload-or-first-write-following-backup-time (TDATE$ format)					
11	disable-flags	zero*				
12	descriptor-flags			reserved	reserved for site use	
13	PCHAR-flags	link-to-section-1-of-main-item				

Table 7-15. Removable Disk Pack Main Item (Sector 0) (continued)

14	*assign mnemonic*			
15	*unused*		*total-nbr-of-times this-file-has-been-assigned*	
16	*unused*			
17	*unused*	*inhibit flags*	*unused*	*absolute-F-cycle-nbr*
18	*date-and-time-last-assignment-ended (TDATE$ format)*			
19	*date-and-time-of-cataloging (TDATE$ format)*			
20	*initial-nbr-of-granules reserved-for-this-assignment*		*nbr-of-granules-of-quota-group-1*	
21	*maximum-nbr-of-granules*		*nbr-of-granules-of-quota-group-2*	
22	*highest-granule-nbr-assigned*		*nbr-of-granules-of-quota-group-3*	
23	*highest-granule-nbr-written*		*nbr-of-granules-of-quota-group-4*	
24	*three-most-significant characters-of-read-key**		*nbr-of-granules-of-quota-group-5*	
25	*three-least-significant characters-of-read-key**		*nbr-of-granules-of-quota-group-6*	
26	*three-most-significant characters-of-write-key**		*nbr-of-granules-of-quota-group-7*	
27	*three-least-significant characters-of-write-key**		*nbr-of-granules-of-quota-group-8*	

* *The format pictured is applicable only to this MFD item as it appears on the removable disk pack.*

Word 10

time-of-first-write following-unload-or-backup-time — If word 12, bit 35 = 1, then word 10 contains the unload time. If word 12, bit 35 = 0, then word 10 contains the time of the first write after the backup file was created or all zeros.

Word 11

disable-flags

Bits 35-32:

1100_2 — Facilities reject (file disabled because it was destroyed)

1010_2 — Facilities warning (file disabled because it was an incomplete write)

	1001_2 SECURE processor reject (file disabled because it was unable to be loaded by the SECURE processor)
Bits 31–0:	Zero (no link to the lead item is necessary since lead items do not exist on removable disk packs).

Word 12

descriptor-flags These bits, when set, indicate:

Bit	
35	Unloaded
34	Backed up
33	Unused
32	Has lapse entries
31	Bad main item sector 1
30	Old item format (created on Executive Level 32 or earlier).
29	Unused
28	Communication between recovery and MFD maintenance routines for removable disk files. Extension sectors of the main item may have been changed on permanent storage. This assures the subsequent update of extension sectors on the disk packs.
27	Removable disk file
26	File is to be made write-only
25	File is to be made read-only
24	File is to be dropped

Word 13

PCHAR-flags These bits, when set, indicate:

Bit	
35	Position granularity
34	Reserved
33	Word addressable
32	Prefers high speed drum

link-to-sector-1 of-main-item Link to sector 1 of main item on removable disk.

Word 17

inhibit-flags These bits, when set, indicate inhibit options on @ASG control statements as follows:

Bit		
29	@ASG,G	Guarded file
28	@ASG,V	Inhibit unload
27	Not @ASG,P	Private file
26	@ASG,X	Exclusive use
25	@ASG,W	Write-only
24	@ASG,R	Read-only

Table 7-16. Removable Disk File Main Item – Sector 1

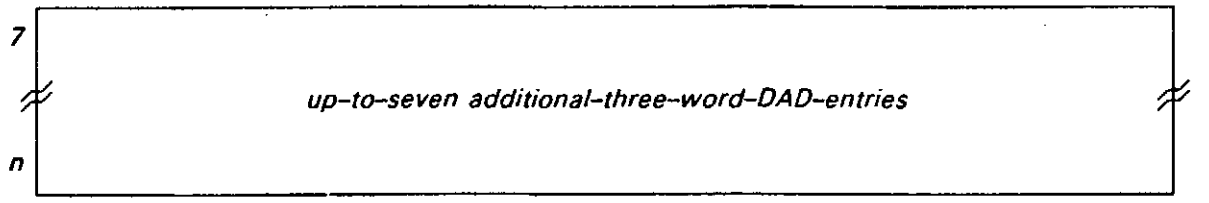
0	U	0	0	0	link-to-sector-2-of-main-item-if-U = 0																		
1																	nbr-of sec-words				reserved		
2	reserved																						
3	reserved-for-security																						
4																							
5																							
6	link-to-sector-0-of-main-item																						
7	nbr-of-backup-reels												nbr-of-two-word lapse-entries								absolute-F-cycle nbr-of-this-file		
8	date-and-time-of-backup-file-creation (TDATE$ format)																						
9	reserved								backup-tape mode-codes				reserved								total-nbr-of-1800-word-text-blocks		
10	tape-noise-constant												starting-file-position of-first-backup-reel								nbr-of-words written-in-last-block		
11	reel-nbr-of-first-reel-of-backup-file																						
12	reel-nbr-of-second-reel-of-backup-file																						

Table 7-16. Removable Disk File Main Item – Sector 1 (continued)

13 14	first-lapse-entry-or-all-zeros
15 16	second-lapse-entry-or-all-zeros
17	unused | nbr-disk-pack-entries
18 19	first-disk-pack-entry
20 21	second-disk-pack-entry
22 23	third-disk-pack-entry
24 25	fourth-disk-pack-entry
26 27	fifth-disk-pack-entry

Table 7-17. Mass Storage File DAD Table

	35	34	33	32	0
0	U	0	0	0	link-to-next-sector-if-U = 0
1	link-to-main-item-or-preceding-DAD-sector				
2	file-relative-address-of-first-word-described-by-this-DAD-table				
3	file-relative-address-of-last-word+1-described-by-this-DAD-table				
4	device-relative-word-address				
5	number-of-contiguous-words-in-this-DAD				
6	DAD-flags				LDAT-index



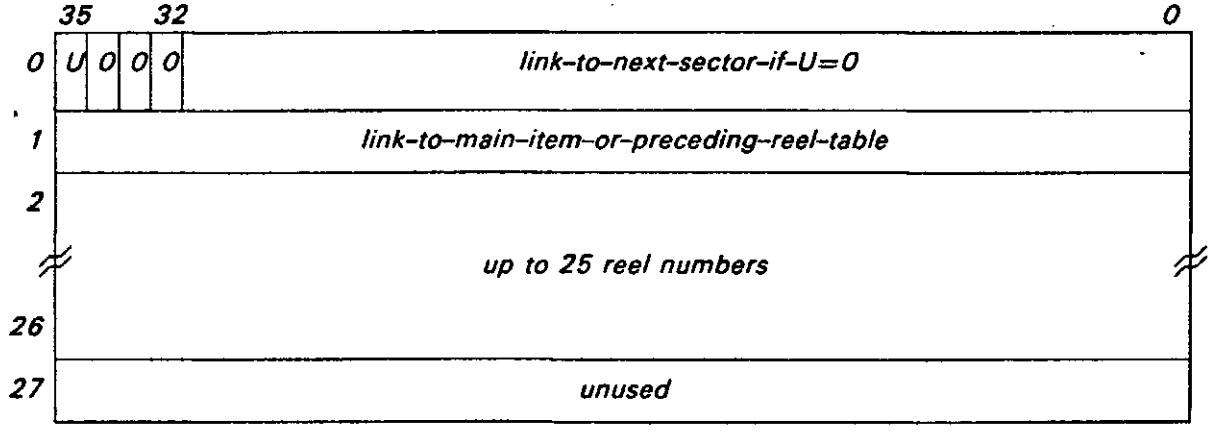
Word 1

When there is more than one DAD table, the second, third, and n^{th} DAD tables are each linked back to the preceding DAD table, and the first DAD table provides the link back to the main item.

Word 6

DAD Flags	01 - DAD has been badspotted 02 - Removable disk 04 - Last DAD in this DAD table
LDAT Index	Index into logical device address table

Table 7-18. Tape File Reel Table



If there are more than 25 reel numbers, then there is more than one reel item. Each reel item lists from 1 to 25 reel numbers. When there is more than one reel item, the second, third, and n^{th} reel items are each linked back to the preceding reel item, and the first reel item provides the link back to the main item.

7.8. F-Cycles

A relative F-cycle number is associated with a particular file, relative to the Master File Directory lead item state at the time of assignment. An MFD lead item contains a link to each F-cycle of a particular file name and directory information that is common to all files in the set. Each link occupies one word in the MFD lead item starting in Word (11 plus number of security words) (see Table 7-8).

For more detailed information on F-cycle philosophy see EXEC Programmer Reference, UP-4144.2 (see Preface).

The positive main item link which is nearest to or actually resides in Word 11 can be referenced as relative F-cycle number -0,0, or +0. The positive main item link which is next nearest to Word 11 can be referenced as relative F-cycle number -1, and so on through the remainder of the lead item.

Therefore, the relative F-cycle number associated with a particular file will change as other files are added to or deleted from the associated F-cycle set.

Figure 7-2 shows the relationship between absolute and relative F-cycle numbers for a particular F-cycle set. For this set, the maximum number of consecutive absolute F-cycles (S3 of Word 9) has been set to 6. The current range of absolute F-cycle numbers (S4 of Word 9) is 5, and the highest absolute F-cycle number (T3 of Word 9) is 5. For the complete lead item format see Table 7-8.

8						
9		04	06	05	0005	
10	0					
11	0	link to absolute F-cycle 5				Link to relative F-cycle 0
12	0	link to absolute F-cycle 4				Link to relative F-cycle -1
13	0	link to absolute F-cycle 3				Link to relative F-cycle -2
14	0	0				
15	0	link to absolute F-cycle 1				Link to relative F-cycle -3
16	0	0				
17	0	0				
18	0	0				
19						

Figure 7-2. Relative Absolute F-Cycle Relationships Example 1

When a new file is being generated, the link is inserted into the proper location in the lead item and the other links are moved down if necessary. The counts in Word 9 are adjusted accordingly. If the file is being created by an @ASG,C or @ASG,U statement, the link will have bit 35 set.

The file becomes cataloged (bit 35 in the link is cleared) when the file is freed or at run termination. When bit 35 is set in the link from the MFD lead item, that file is not considered in the relative cycle numbering scheme and therefore it is not included in S2 of Word 9 (number of cataloged F-cycles in the lead item). For example, if one adds two files to the lead item shown above by @ASG,C FN(7) and an @ASG,C FN(+1), the lead item is as shown in Figure 7-3.

This sequence has pushed absolute F-cycle 1 out of the maximum allowable F-cycle range (S3 of Word 9). Therefore the system deleted absolute F-cycle 1. If F-cycle 1 had been currently assigned, the system would have been unable to delete it, so the @ASG,C FN(7) would have received an F-cycle conflict (FAC REJECTED; with bit 5 set in the status word).

The fact that F-cycle +1 is currently assigned prevents further cataloging within this cycle set. Bit 34 in word 10 indicates this situation exists. When the +1 file is freed it will become relative F-cycle 0 and further cataloging will be allowed. An attempt to catalogue a new file within the set when +1 file is being created will cause the F-cycle conflict FAC REJECTED; i.e., if the preceding example had specified the @ASG,C FN(+1) and then a @ASG,C FN(7), the @ASG,C FN(7) would be rejected.

There is one other restriction on F-cycle generation that will cause an F-cycle conflict FAC REJECTED. For a new cycle to be created, its absolute F-cycle number must be within the following range:

$$(x-w) < z \leq (x-y+w+1)$$

where:

- *z* absolute F-cycle number requested
- *w* S3 of Word 9 of the lead item (maximum number of F-cycles)
- *x* T3 of Word 9 of the lead item (cycle number of latest F-cycle)
- *y* S4 of Word 9 of the lead item (current range of F-cycles)

This range prevents more than one file from being dropped from the set (the oldest cycle) when a new file is being added to the set.

8			*w*	*y*	*x*	
9		03	06	06	0010	
10	2					
11	4	link to absolute F-cycle 010				Link to relative F-cycle +1
12	4	link to absolute F-cycle 7				
13	0	0				
14	0	link to absolute F-cycle 5				Link to relative F-cycle 0
15	0	link to absolute F-cycle 4				Link to relative F-cycle -1
16	0	link to absolute F-cycle 3				Link to relative F-cycle -2
17	0	0				
18	0	0				
19						

Figure 7-3. Relative Absolute F-Cycle Relationships Example 2

7.9. SECURE Tape Backup Copies

The function of the SECURE processor is to protect the physical security of cataloged files, whether they reside on mass storage or removable disk, by creating backup copies on tape. These tapes are an integral part of the sophisticated cataloged file recovery process available through SECURE.

For more detailed information on SECURE, see *Series 1100 File Administration Processor (SECURE), Programmer Reference*, UP-8726 (applicable version.)

The tape format for SECURE is described in 7-4, 7-5, and 7-6.

Figure 7-4. SECURE Tape Block 1

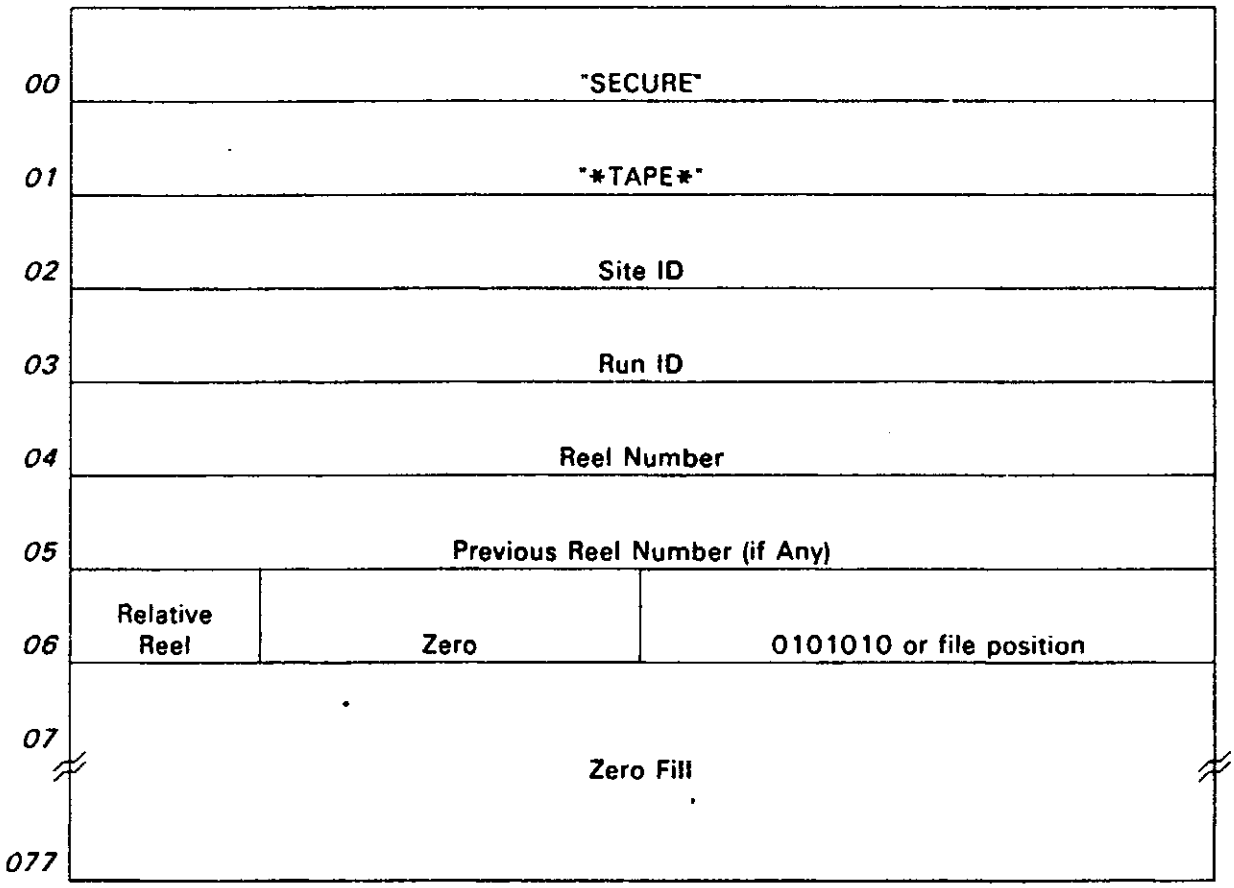


Figure 7-5. SECURE Tape Directory Item Block Format

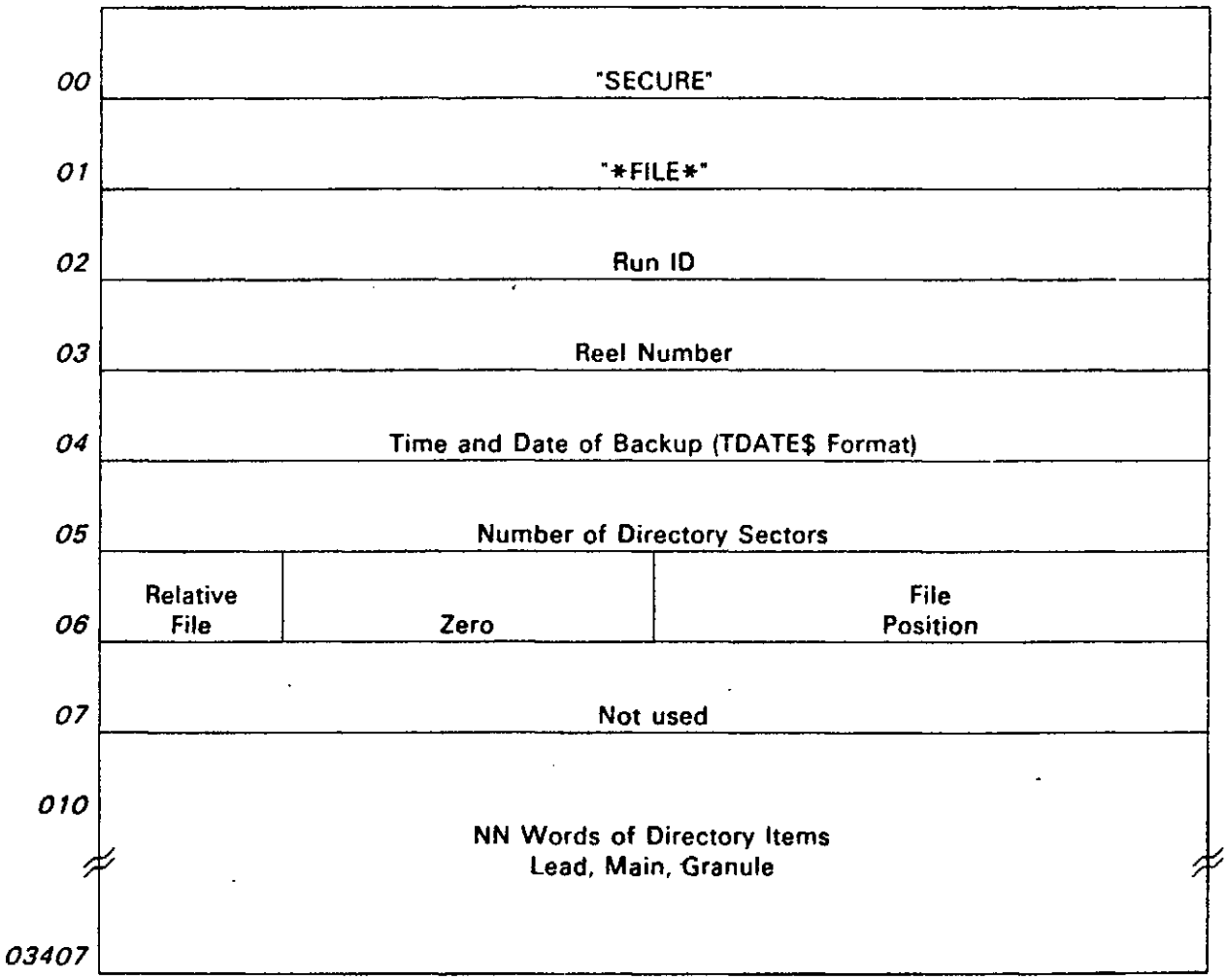
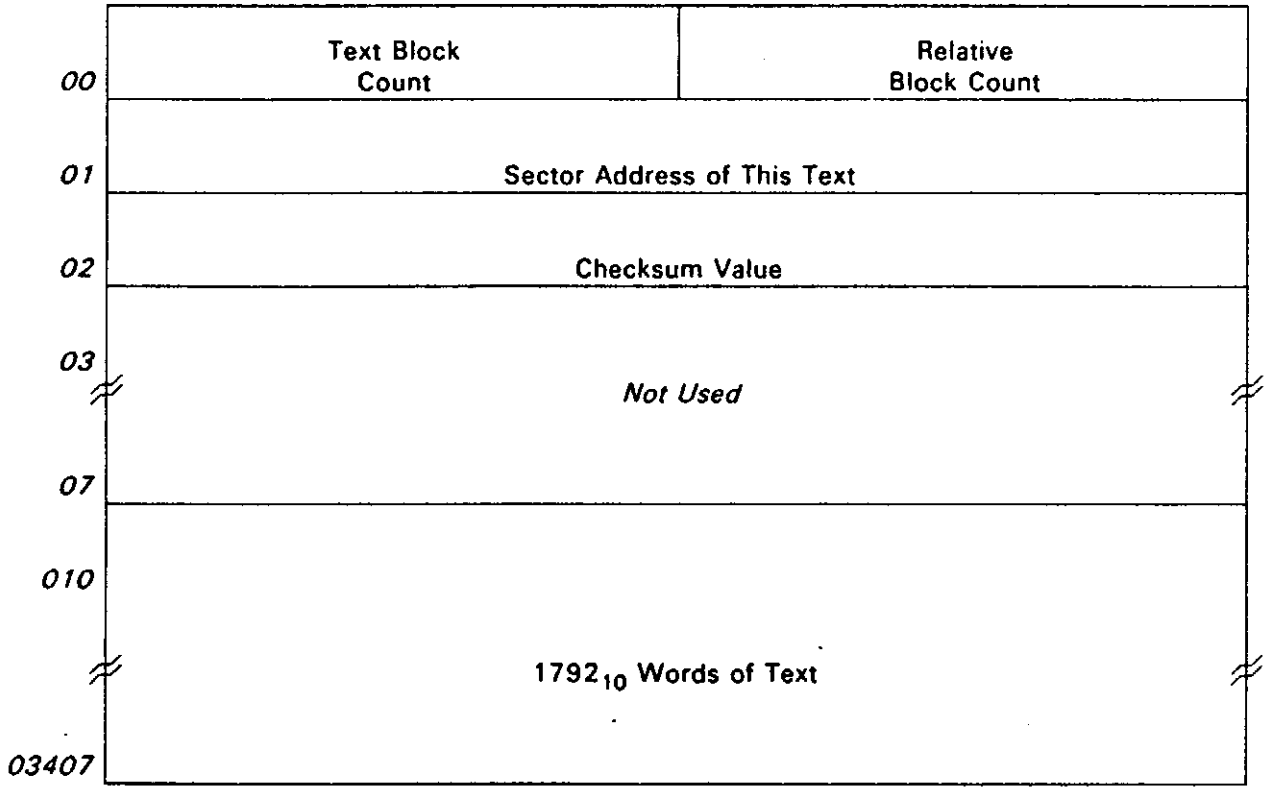


Figure 7-6. Tape Text Track Block Format



Text Block Count starts at zero and is incremented by one for every block copied on the tape for the file.

Relative Block Count refers to the track address of the file when it was saved.

Bit 35 of Word 0 is set for the last block of a given file.

The end-of-reel label is written if there is another tape to follow. The label consists of 05416 in T1 of Word 0 of a 14 word block.

The end-of-tape label is written if no more is written on a tape and there is no tape following (for the file being written). The label consists of "EOTEOT" in Word 0 of a 14 word block and is terminated by two end-of-file marks.

7.10. System Files

This section describes files SYS$*RUN$, the general EXEC file (GENF$), SYS$*PRINTER$, and RUN$, RLIB$, and PRINTER$ file protection.

7.10.1. SYS$*RUN$

The file SYS$*RUN$ is cataloged by the canned run, SYS, during a boot with Select Jump switch 4, or Select Jump switches 4 and 13 set. SYS$*RUN$ is the 5th file on the boot tape which is in COPY,G format. RUN$ is a program file that consists of a number of symbolic and absolute elements.

The one element that must be present in this file is the symbolic element called BOOTELT. This element is a partial runstream added (@ADD) and executed as part of the SYS run. Its functions in the SYS run are:

1. to catalog and load SYS$*RLIB$ from the 6th file on the boot tape if a Select Jump switch 4 or a Select Jump switch 4 and 13 boot is taking place,

2. delete the old SYS$*SYS$MAP and catalog and load from the 2nd or 7th file on the boot tape a new SYS$*SYS$MAP if a tape boot is taking place,

3. call SECURE to list all disabled files if a recovery boot is taking place, and

4. start the element SYS$*RUN$.PRINTER$, if this is an I-boot, to load SYS$*PRINTER$.

The determination of the type of boot taking place is made by testing the condition word that is set-up by a @SETC control statement in the beginning of the SYS run. The SETC control statement is one of the "canned" control statements existing for the SYS run which exists in the element INDRIV. Before starting the SYS run, INDRIV modifies the canned SETC control statement by setting up the "value" part of the control statement to reflect the type of boot operation taking place. The following is a list of the condition word values and their associated meanings:

1. S3
   - 00 Drum disk boot
   - or
   - 01 Tape boot

2. S4
   - 02 Jump key switch 13 and 4 boot
   - 01 Jump key switch 4 boot
   - 00 Neither Jump key switch 4 or 13 set.

Also in INDRIV is the "canned" image used by SYS to assign (create) the file SYS$*RUN$. This image is:

```
@ASG,CPG SYS$*RUN$//RUNWR$.,F2/1//10000
```

although in previous releases the image:

```
@ASG,CPG SYS$RUN$//RUNWR$,F2
```

was used. In the later case, the maximum size to which the file could grow defaulted to the system standard value (MAXGRN) which could be made so small by the user (SITE) that the system could not be booted.

Other elements in SYS$*RUN$ are:

1. LOGFED

   the absolute form of LOGFED.

2. LOGFED

   a runstream that contains a call to LOGFED.

3. FIREUP

   a runstream that catalogs and loads the 1100 Test Package tests.

4. DPREP

   a runstream that initiates a disk prep.

5. DLLLOD

   a runstream that initiates Down Line Load.

6. DLLDMP

   a runstream that initiates SSP Dump Save.

7. PRINTER$

   a runstream to load SYS$*PRINTER$.

7.10.2. General EXEC File (GENF$)

SYS$*GENF$ is a general EXEC file, and it is a cataloged file, thus allowing recovery of the file. The file contains the following items.

1. Symbiont output queue
2. The scheduling queue
3. The temporary log queue
4. Facility INFOR statements for batch runs which are presently undergoing facility synopsis or which are in a facility wait state.
5. Spooled output for demand runs.

7.10.3. SYS$*PRINTER$

The file SYS$*PRINTER$ contains absolute elements required to use the 0768, 0770, and 0776 printers. This file also contains user-specified load codes. The elements normally in this file are the following absolute elements:

STANDARD768 The standard load codes for the 0768 printer generated by the symbolic element STANDARD768.

STANDARD770 The standard load codes for the 0770 printer generated by the symbolic element STANDARD770.

STANDARD776 The standard load codes for the 0776 printer generated by the symbolic element STANDARD776.

The site may place additional load codes in any absolute element in the file SYS$*PRINTER$ using the format described in EXEC Programmer Reference, UP-4144.2 (see Preface.)

It is advisable for the site to limit write access to this file since changing the data in the elements can result in unreadable printouts and operator messages.

This file must have the appropriate absolute element in it (STANDARD768 for 0768 printers, STANDARD770 for 0770 printers, STANDARD776 for 0776 printers) and be assignable (rolled in and not hardware disabled or exclusively assigned) if 0768, 0770, or 0776 printers are to be used. The site may modify the standard load codes by replacing the standard absolute elements.

Sites that have user load code elements should specify a procedure to ensure that user elements are added to the library tapes used for system generations. These sites would normally build the file PRINT$UTIL using the procedure described in the EXEC System Generation User Guide, UP-8448 (see Preface) and then copy the user elements into PRINT$UTIL and replace the printer libraries on their LIB tape before generating the boot tape.

8. Tape Labeling System

This subsection defines the tape labeling system. This system incorporates the American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1969, to provide a comprehensive tape labeling facility.

The intent of a tape labeling facility is twofold. First, an installation may desire tape labeling to minimize the impact of operator and user error, either intentional or unintentional. Second, a site may wish to facilitate a smooth data transfer between computing systems via magnetic tape. These computing systems may be logically close (may be under common managerial control) or they may be quite apart. In either case, a need exists to provide a functional interface between data on one system and data on a different system. In this instance, data interchange is the primary objective of a tape labeling facility.

One of the primary objectives of tape labeling is to provide a way that will allow a specific site to define its position with respect to the extremes of data integrity/security and data interchange. The first step in providing this facility is defined in this section as a 'common ground'. This 'common ground' is referred to as the base level throughout this section.

The base level of the tape labeling facility in the Operating System delivered to a site creates, maintains and validates certain label records within the constructs of the American National Standard. Other optional label records can be processed depending on site implementation.

8.1. Definitions

This section provides definitions of the terms used in connection with tape labeling and the labels used by both the system and the user.

"a"

One of the following characters: the digits 0,1,...,9, the upper case letters A,B,...,Z, and the following special characters:

(Space) ! " % & ' ( ) * + , - . / : ; < = > ?

The ASCII characters were chosen from the center four columns of the code table specified in ASCII, except for those positions where there is a provision for alternative graphic representation.

Block

A group of contiguous characters recorded on and read from magnetic tape as a unit. A block may contain one or more complete records.

Double Tape Mark

A delimiter, consisting of two tape marks (see definition below), that is used to indicate the end of a volume or of a file set. It also occurs when an empty file section or an empty file exists on a volume, in which case they are not interpreted as a double tape mark but rather as two single tape marks framing an empty file section. "Empty" here means that no blocks are present between the tape mark following the beginning-of-file section label group and the tape mark preceding the end-of-file section label group or end-of-file label group of that file section or file.

File

A collection of information in data blocks, consisting of records pertaining to a specified subject and delimited by tape marks and label groups. The absence of information or data blocks results in the creation of a null or empty file, that is, a file without information.

File Section

That part of a file recorded on any one volume. The sections of a file may not have sections of other files interspersed.

File Set

A collection of one or more related files recorded on one or more volumes. A file set may consist of:

- One file recorded on a single volume
- More than one file recorded on a single volume.
- One file recorded on more than one volume.
- More than one file recorded on more than one volume

Header Label Group

A label group consisting of a system volume header label (if it is the first label group on the volume), system file header labels, and a tape mark. Optionally and under the control of the user, the group may also contain user file header labels following the appropriate system labels.

Label

An 80 character block at the beginning or end of a volume or file serving to identify and/or delimit that volume or file. It is not considered part of the file section it delimits.

Label Group

A collection of contiguous labels pertaining to a file which precedes or follows that file or part of that file on a volume. The volume header label(s) and the following file header label(s) are considered to be the first label group on the volume.

Label Identifier

A name consisting of three alphabetic characters identifying the type of label. It is followed by a label number.

Label Number

A numeric character in the range 1 to 9 following the label identifier.

"n"

Any numeric digit in the range 0 through 9.

System Labels

Those labels controlled by the Operating System and which the user may not access directly. System end-of-volume and system end-of-file labels are mutually exclusive in any given end label group. The system labels are:

- VOL1 Volume header label
- HDRn File header label. Must be in consecutive ascending order and correspond in number to EOF/EOV labels
- EOFn End-of-file label (corresponds to HDRn with the same rules applying)
- EOVn End-of-volume label (corresponds to HDRn with the same rules applying)

Tape Mark

A special code configuration or sequence recorded on magnetic tape indicating the boundary between files and labels and also between certain label groups. The presence of the tape mark is generally detected by the tape unit or by the tape unit controller and is thereby signalled to the Executive. (See the section on double tape marks).

User Labels

Those labels whose reading, writing, and contents (except for the initial label identifier and number) are controlled by the user. They are:

- UHLa User file header label. No order or correspondence of occurrence is necessary
- UTLa User Trailer Label. Also known as user end-of-volume or user end-of-file labels, depending on the circumstances. (No correspondence to UHLa is necessary.)

Volume

A physical unit of storage media. In this specification the medium is magnetic tape, so a volume is a reel of magnetic tape.

8.2. Structure of Magnetic Tape File

SINGLE FILE, SINGLE VOLUME

VOL1 HDR1 HDR2*—file A—*EOF1 EOF2**

SINGLE FILE, MULTIVOLUME

VOL1 HDR1 HDR2*—first part file A—*EOV1 EOV2**

VOL1 HDR1 HDR2*—last part file A—*EOF1 EOF2**

MULTIFILE, SINGLE VOLUME

VOL1 HDR1 HDR2*—file A—*EOF1 EOF2* HDR1 HDR2*—file B *EOF1 EOF2**

MULTIFILE, MULTIVOLUME

VOL1 HDR1 HDR2*—file A—*EOF1 EOF2*HDR1 HDR2*—first part file B—*EOV1 EOV2**

VOL1 HDR1 HDR2*—continuation of file B—*EOV1 EOV2**

VOL1 HDR1 HDR2*—last part of file B—*EOF1 EOF2*HDR1 HDR2*—file C—*EOF1 EOF2**

NOTE: An asterisk (*) indicates one tape mark and two asterisks (**) indicate two tape marks.

8.3. Format of Labels

The following subsections depict volume header labels, first file headers, second headers, first end of file labels, second end of file labels, and optional labels.

8.3.1. Volume Header Label (VOL1)

Character Position	Field Nbr	Contents	Length (Char)
1–3	1	VOL	3
4	2	1	1
5–10	3	Reel Number	6
11	4	Accessibility Code	1
12–31	5	Spaces	20
32–37	6	Spaces	6
38–51	7	Owner Identification	14

Character Position	Field Nbr	Contents	Length (Char)
52–79	8	Spaces	28
80	9	Label Standard Version	1

NOTE: All characters are ASCII.

8.3.2. First File Header (HDR1)

Character Position	Field Nbr	Contents	Length (Char)
1–3	1	HDR	3
4	2	1	1
5–21	3	File Identifier	17
22–27	4	Set Identification	6
28–31	5	File Section Number	4
32–35	6	File Sequence Number	4
36–39	7	Generation Number	4
40–41	8	Generation Version Number	2
42–47	9	Creation Date	6
48–53	10	Expiration Date	6
54	11	Accessibility	1
55–60	12	Block Count	6
61–73	13	System Code	13
74–80	14	Reserved for Future Standardization	7

8.3.3. Second Header (HDR2)

Character Position	Field Nbr	Contents	Length (Char)
1-3	1	HDR	3
4	2	2	1
5	3	Record Format	1
6-10	4	Block Length	5
11-15	5	Record Length	5
16-50	6	Reserved for Operating Systems	35
51-52	7	Buffer Offset	2
53-80	8	Reserved for Future Standardization	28

8.3.4. First End of File Label (End of Volume)

Character Position	Field Nbr	Contents	Length (Char)
1-3	1	EOF (EOV)	3
4	2	1	1
5-54	3-11	Same as Corresponding Fields in First File Header Label	50
55-60	12	Block Count	6
61-80	13-14	Same as Corresponding Fields in First File Header Label	20

8.3.5. Second End-of-File Label (Second End-of-Volume)

A second end-of-file label (end-of-volume) contains EOF2 (EOV2) as the label identifier and label number and contains the same information as fields three through eight of the HDR2 label.

8.3.6. Optional Labels

Character Position	Field Nbr	Contents	Length (Char)
1-3	1	UHL or UTL	3
4	2	Label Number	1
5-80	3	User Option	76

8.4. Label Field Definitions

The following subsections cover various label field definitions. These labels are: volume header, system file header, user file header, system end-of-file, system end-of-volume, and user trailer.

8.4.1. Volume Header Label Set

The volume header label set is composed of one label record, the VOL1 record. This record is created, maintained, and verified by the base level (see 8.3.1) and is always the first record of a volume.

Character Position: 1 to 3

Field Name: Label Identifier

Length: 3 Characters

Description:

The contents of this field are always VOL. This string is the first indication to the EXEC that the volume is labeled. The contents of this field are verified, created, and maintained by the base level.

Character Position: 4

Field Name: Label Number

Length: 1 digit

Description:

The contents of this field are always 1. This string is the second indication to the EXEC that the volume is labeled. Label records with a label identifier of VOL and a label number other than 1 are prohibited by the base level.

Character Position: 5 to 10

Field Name: Volume Identifier

Length: 6 "a" characters

Description:

The contents of this field are assigned by the owner of this volume to distinguish it from other volumes under the owner's control. The base level maintains and verifies this field. The contents are created by the base level using information provided by the site. The base level requires this field to contain the reel number that corresponds with that used on the user's @ASG statement.

Character Position: 11

Field Name: Accessibility

Length: 1 "a" character

Description:

The contents of this field indicate restrictions, or the lack thereof, on access to the information in this volume. The base level defines two values for the contents of this field. One value (a space) indicates unlimited accessibility. The other value (a comma ",") restricts access of this volume to users with the same account number as that contained in the owner identification field of this label record. All other values are treated as a space by the base level. Values reserved for site purposes are: A B C D E F G H I J K L ! " $ % & '

Character Position: 12 to 37

Field Name: (reserved for future standardization)

Length: 26 spaces

Description:

The base level does not place any characters other than spaces in this field. If upon verification the EXEC notes non-spaces in this field, it ignores them.

Character Position: 38 to 51

Field Name: Owner Identification

Length: 14 'a' characters

Description:

The contents of this field identify the owner of the volume. The base level normally expects spaces in this field. If the accessibility field of this record contains a comma, this field is expected to contain the account number of the owner of this volume. If the accessibility field does not contain a comma, the contents of this field are ignored by the base level.

Character Position: 52 to 79

Field Name: (reserved for future standardization)

Length: 28 spaces

Description:

The base level inserts spaces in this field. On verification, the base level ignores this field.

Character Position: 80

Field Name: Label Standard Version

Length: 1 digit

Description:

The contents of this field indicate the Standard to which this volume conforms. A "1" indicates the American National Standard.

8.4.2. System File Header Label Set

The system file header label set is created, verified, and maintained by the base level. The base level maintains two records of this label set, the HDR1 and HDR2. Other HDR labels are recognized but ignored by the base level.

8.4.2.1. The HDR1 Label Record

Character Position: 1 to 3

Field Name: Label Identifier

Length: 3 characters

Description:

The string HDR must occur in this field for the base level to recognize this record as part of the system file header label set.

Character Position: 4

Field Name: Label Number

Length: 1 digit

Description:

The base level recognizes any valid label number, but it only processes and creates records whose label number is either "1" or "2". All others are ignored.

Character Position: 5 to 21

Field Name: File Identifier

Length: 17 "a" characters

Description:

This field's contents identify the file to its owner. The base level creates, verifies, and maintains the contents of this field. The base level, in the normal case, inserts and verifies a string which identifies the qualifier and filename used on the user's @ASG statement that created this file. If the number of character positions of the qualifier and filename (the sum of these two) exceed 16 positions, the right most portion of the qualifier is truncated, but the entire filename is always maintained. For example, if the actual qualifier and filename used was: MYPROJECTID*CREATEDFILE, the base level inserts MYPRO*CREATEDFILE into this field. In the default case, the same character string occurs in this field for all files within a file set. In other words, all HDR1 records for all files on a volume or multiple volumes contain the same file identifier in the default case. The file identifier in the default case is left justified, space filled.

Character Position: 22 to 27

Field Name: File Set Identifier

Length: 6 "a" characters

Description:

This field identifies this file set among other file sets. The base level inserts, on output, the information from the volume identifier field of the VOL1 record of the first or only volume of the file set into this field. The contents of this field are not verified by the base level.

Character Position: 28 to 31

Field Name: File Section Number

Length: 4 'n' digits

Description:

This field identifies this section among the sections of this file. The base level inserts the correct number into this field in the case of multiple volume files. No verification is performed upon this field.

Character Position: 32 to 35

Field Name: File Sequence Number

Length: 4 'n' digits

Description:

This field identifies the placement of this file among the other files of this file set. The file sequence number of the first file of a set is 1 and it is incremented by 1 for each additional file. The base level creates and maintains the contents of this field.

Character Position: 36 to 39

Field Name: Generation Number

Length: 4 "n" digits

Description:

This field distinguishes among successive generations of this file. The base level inserts, on output, the absolute F-cycle of the file into this field. The base level does not verify the contents of this field.

Character Position: 40 to 41

Field Name: Generation Version Number

Length: 2 "n" digits

Description:

The contents of this field are to be used to distinguish successive iterations of the same generation. As this field has no significance to the Series 1100 Operating System, it is ignored by the base level on input. On output, the value of zero is inserted.

Character Position: 42 to 47

Field Name: Creation Date

Length: 6 characters

Description:

The contents of this field indicate the date upon which this file was created. The field's construction is a space followed by two "n" digits for the year and three "n" digits indicating the day within the year. This field is maintained and created by the base level.

Character Position: 48 to 53

Field Name: Expiration Date

Length: 6 characters

Description:

The contents of this field indicate the date upon which this file is considered expired by its owner. The field's construction is the same as for the creation date field. The base level creates (using user provided information), maintains, and verifies this field.

Character Position: 54

Field Name: Accessibility

Length: 1 "a" character

Description:

The contents of this field indicate restrictions, or the lack thereof, on access to the information in this file. The base level provides a limited number of predefined values for this field. This field is created, maintained and verified by the base level. The base level defines three definitions for this field. These definitions are not mutually exclusive. Any, none, or all of these definitions may be used by a user. Bit 0 indicates that the verification of the file identifier field of this record is required. Bit 1 indicates the file to be a WRITE ONLY file. Bit 2 indicates this file to be a READ ONLY file. This value, or combination of values, is added to 060 to form accessibility characters 061–067. Any value not recognized by the base level will result in the following: (1) File identifier verification not required, (2) file is not READ ONLY, (3) file is not WRITE ONLY. Values reserved for site use are: A B C D E F G H I J K L ! " $ % & '

The above values of 061–067 will be determined by the R, W, and F options on the assign card; i.e., if the R option is on the assignment when the file is created, the file is read only until it expires. Table 8–1 shows HDR1 Accessibility codes.

Table 8–1. HDR1 Accessibility Codes

Accessibility Character	Assign Options		
	F	W	R
040	X		
061			
062	X	X	
063		X	
064	X		X
065			X
066	X	X	X
067		X	X

Character Position: 55 to 60

Field Name: Block Count

Length: 6 zero digits

Description:

The purpose of this field is to maintain a correspondence with the same field in the EOF1 and/or EOV1 record. The base level creates this field with zeros and maintains this value.

Character Position: 61 to 73

Field Name: System Code

Length: 13 "a" characters

Description:

The contents of this field indicate the system that recorded this file. The left most 7 positions of this field are created, maintained, and verified by the base level. These 7 positions are used to identify label records written by the base level and the update version to the base level. The remaining 6 positions are reserved for use by the site to identify its tape labeling facility implementation. In the base level, the left most characters are "U1100-1".

Character Position: 74 to 80

Field Name: (reserved for future standardization)

Length: 7 spaces

Description:

The base level inserts spaces in this field. On verification, the base level ignores this field.

8.4.2.2. The HDR2 Label Record

Character Position: 1 to 3

Field Name: Label Identifier

Length: 3 characters

Description:

The string HDR must occur in this field for the base level to recognize this record as part of the system file header label set.

Character Position: 4

Field Name: Label Number

Length: 1 digit

Description:

To be recognized as a HDR2 record by the base level, this field must contain a "2".

Character Position: 5

Field Name: Record Format

Length: 1 "a" character either F, D, S, or U

F = Fixed Length
D = Variable with number of characters in the record specified in decimal.
V = Variable with the number of characters specified in binary.
U = Unspecified

Description:

This field defines the record format of the data file. As applied to Series 1100 Systems, this field has little significance. On output, in the default case, the base level enters a "U" in this field. On input, the base level ignores the contents of this field. A user execution level facility (ER TLBL$) is provided in the base level to allow a user to provide one acceptable value for this field. On input, the facility allows a user to retrieve the contents of this field.

Character Position: 6 to 10

Field Name: Block Length

Length: 5 "n" digits

Description:

The field specifies the maximum number of characters per block of the data file. As applied to Series 1100 Systems, this field has little significance. On output, in the default case, the base level inserts the maximum possible value in this field (99999). On input, this field is ignored. A user execution level facility (ER TLBL$) is provided in the base level to allow a user to provide a valid value for this field. This facility also allows a user to retrieve the contents of this field.

Character Position: 11 to 15

Field Name: Record Length

Length: 5 "n" digits

Description:

This field defines the logical record length of records in the data file. As applied to Series 1100 Systems, this field has little meaning. On output, in the default case, the base level inserts zeros into this field. A user execution level facility (ER TLBL$) is provided in the base level to allow a user to provide a valid value for this field. This facility also allows a user to retrieve the contents of this field.

Character Position: 16 to 50

Field Name: (reserved for system use, of which positions 16–32 are reserved for site use.)

Character Position: 51 to 52

Field Name: Buffer Offset Length

Length: 2 "n" digits

Description:

This field specifies the length in characters of any additional field inserted before the first record in a data block. As applied to Series 1100 Systems, the field has little significance. On output, in the default case, the base level inserts zeros in this field. On input, this field is ignored by the base level. A user execution level facility (ER TLBL$) is provided by the base level to allow a user to provide a valid value for this field. This facility also allows a user to retrieve the contents of this field.

Character Position: 53 to 80

Field Name: (reserved for future standardization)

Length: 28 spaces

Description:

The base level inserts spaces in this field on output. On verification, the base level ignores this field.

8.4.3. Header Labels 3-9

The base level does not provide for the implementation of HDR 3-9 labels. The base level bypasses them during label checking if they exist.

8.4.4. User File Header Label Set

The records that compose the user file header label set are those label records whose label identifier is the string UHL. They are the responsibility of the user. The base level provides an execution level facility (ER TLBL$) for a user to create and process records from this label set. On output, the base level insures that two fields (the label identifier and label number field) contain valid values. The correct placement of these records after the system file header label set is also insured by the base level. On input, these records are ignored by the base level.

Character Position: 1 to 3

Field Name: Label Identifier

Length: 3 characters

Description:

The base level verifies the user has provided the string UHL in this field during creation.

Character Position: 4

Field Name: Label Number

Length: 1 digit

Description:

The only allowable values for this field are 1 through 9. The base level verifies the validity of this field. The order of the records using this field is not verified.

Character Position: 5 to 80

Field Name: (reserved for user application)

Length: 76 "a" characters

Description:

The user may provide any "a" character information desired in this field.

8.4.5. System End–of–File Label Set

The system end–of–file label set is composed of label records whose label identifier field contains the string EOF. The base level supports two records from this set: the EOF1 and EOF2 records. The EOF1 is an exact duplicate of the HDR1 record of the file with the exception of the label identifier and block count fields. The block count field of the HDR1 record is always "00000". The block count field of the EOF1 record has exactly the same character position and length as the field in the HDR1, but it contains the number of data blocks since the beginning of the data file. The base level validates this field on input and logs any error to the user.

A user execution level facility (ER TLBL$) is provided to allow a user to provide and retrieve the contents of this field.

The EOF2 record is an exact duplicate of the HDR2 record of the file with the exception of the label identifier field.

The EOF label set is used primarily during READ BACKWARD operations. The base level performs file verification using the HDR label set if the user is attempting WRITE or READ FORWARD operations. The EOF label set is used during file verification if the user is attempting READ BACKWARD operations.

8.4.6. System End–of–Volume Label Set

The system end–of–volume label set is composed of label records whose label identifier field contains 'EOV'. The base level supports two records from this label set: EOV1 and EOV2. The discussion of the system end–of–file label set also applies to this label set. The format and description are identical to those of the end–of–file label records. The EOV labels are used to signal a tape swap is necessary.

8.4.7. User Trailer Label Set

The records that compose the user trailer label set are those label records whose label identifier consists of the string UTL. They are the responsibility of the user. The discussion of user file header label records above is also applicable to the discussion of user trailer label records.

9. Log Entry File Formats

Each value in a master log file entry is binary unless it is obviously a symbolic name such as program name, version name, run-id, qualifier, filename, project-id, or account number, in which case it is in Fieldata that is left-justified and space filled. Several items in all entries have a common format and are described here to facilitate easier understanding of the following entry formats. The common items are as follows:

- date and time

  This is the format of time and date as received from a call to TDATE$, (i.e., TDATE$ format).

- message

  Refers to a string of Fieldata characters, the maximum length of which is specified in the entry format.

- system indicator

  An indicator used by log editing programs to distinguish changes in log file formats produced by different levels of the Executive.

- nbr-of-log-entries

  Used only for the first entry in a 224-word block.

- nbr-of-words-in-entry

  A count of words used in the entry exclusive of Words 0, 25, 26 and 27.

Each of the entry formats contains a word that provides either the date and time of the log entry or the date and time of some program action. The format of this word is:

mm/dd/yy/seconds-after-midnight

where the year (yy) is modulo 64.

The log file contains a log file header created on the initial access to the log file.

Figure 9-1 illustrates the format of the log file header. Note that it is organized into two sections; a Table of Contents (TOC) section and a table section. The descriptive information in the header is contained within the tables where each table contains a discrete set of information.

The TOC section contains the following information:

1. The identifier word "SYSLOG",
2. Sector address of first log entry,
3. Number of tables, and
4. Two word descriptors for each table.

Each descriptor contains a Fieldata table name, the starting address of the table and the length of the used portion of the table. The starting address is a word address which is relative to the start of the header. The table length is specified in number of words. The TOC is currently limited to 224 words or 110 tables.

At present, only the equipment mnemonic table is defined. The Fieldata table name for this table is "EQPMNM".

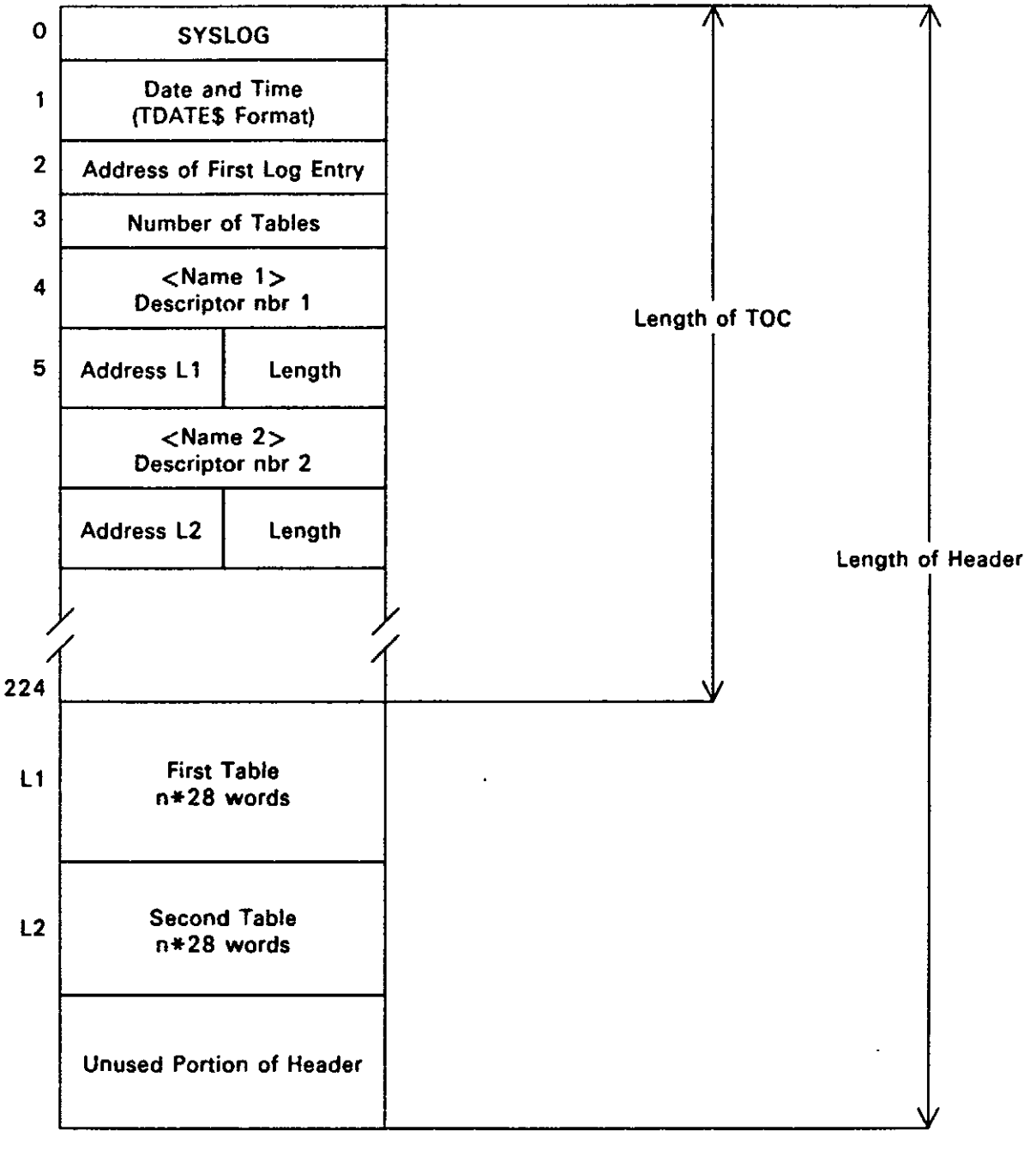


Figure 9-1. Log File Header Format

9.1. Control Statement Log Entries

Log entries specified by the @LOG control statement are placed in the master log file in the order in which they occur. The entry format is:

Table 9–1. Control Statement Log Entries

0	*entry–type (1)*	*nbr–of–wrds in–entry*	*system indicator*	*reserved*	*reserved*	*nbr–of–log entries–in 224–wrd–blk*
1	*message (22–word maximum)*					
24						
25	*date–and–time–of–log–entry*					
26	*reserved*					
27	*run–id*					

9.2. Facility Usage Log Entries

Whenever the configuration of a run is changed by assigning a tape or arbitrary device file, an entry is made in the master log file. The entry format is:

Table 9–2. Facility Usage Log Entries

0	*entry–type (2)*	*nbr–of–wrds in–entry*	*system indicator*	*reserved*	*reserved*	*nbr–of–log entries–in 224–wrd–blk*
1	*equipment code*		*reserved*	*TODRUM–addr–for–assign/free of–a–communications–device*		
2	*logical–device–name*					
3	*entry–2 (same format)*					
4						

Table 9-2. Facility Usage Log Entries (continued)

22 23	*entry-11*
24	*SUPs + voluntary-delay-time*
25	*date-and-time-of-log-entry*
26	*reserved*
27	*run-id*

NOTES: 1. *For a communications device assign the logical device name replaced by the TODRUM address.*

2. *Words 1-2 are for entry 1, Words 3-4 are for entry 2, ..., Words 21-22 are for entry 11. All entries have the format shown for entry-1.*

9.3. Cataloged Mass Storage File Usage Entry

Whenever a cataloged file is created, assigned, or freed (using a @FREE control statement or at run termination) a log entry is created and subsequently inserted into the master log. The entry format is:

Table 9-3. Cataloged Mass Storage File Usage Entry

0	*entry-type (3)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1 2	qualifier					
3 4	filename					
5 6	project-id					

Table 9-3. Cataloged Mass Storage File Usage Entry (continued)

7	account-number			
8				
9	unused	flag 2	current-assigns	absolute-F-cycle
10	date/time-of-FREE-or-0			
11	*date/time-of-cataloging*			
12	*date/time-of-ASG-or-zero*			
13	*Words 13 through 20 contain count of file granules at time of log entry creation which exist on mass storage devices having equipment codes 030 through 037.*			
20				
21	*equipment-code*	*unused*		
22	*unused*			
23	*unused*			
24	*SUPs + voluntary-delay-time*			
25	*date-and-time-of-log-entry*			
26	*reserved*			
27	*run-id*			

Word 9

current assigns — The number of current assignments for this file when the log entry is created.

flag 2

040	Position granularity
020	Private file
010	File is being dropped
004	File was or is being assigned with exclusive use
002	Write only file
001	Read only file

9.4. Program Termination Log Entry

For each program in the run, termination information is entered in the master log. The entry format is:

Table 9-4. Program Termination Log Entry

0	*entry-type (4)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1 2	*program-name*					
3 4	*version-name*					
5	*program-termination-date/time*					
6	*CPU-time*					
7	*quantum-timer-value-in-100-nanoseconds-intervals (1100/80)* or quantum-timer-value-in-116-nanoseconds-intervals (1100/60)*					
8	*ER-and-control-card-charge (200 microsecond intervals)**					
9 ≈ 18	*Words 9 through 18 contain I/O transfer costs for Group 1 through Group 10 I/O devices**					
19	*SUPs (200 microsecond intervals)**					
20	*CBSUPs (core-block-SUPs)**					
21	*voluntary-delay-time (200 microsecond intervals)**					
22	*time-in-real-time-mode (200 microsecond intervals)**					
23	*unused*			*last-reentry-address*		
24	*condition-word*					
25	*date-and-time-of-log-entry*					

Table 9-4. Program Termination Log Entry (continued)

26	*reserved*
27	*run-id*

* *The values for CPU time, SRC counts, ER and control card charges, in the Type 4 and the Type 16 log entries reflect what has accumulated in the run up to the time of the log entry is built. In computing the quantity used in the execution of a program, it is necessary to find the differences between the corresponding values in the Type 4 and Type 16 log entries for the program.*

9.5. Run Termination Log Entry

At the completion of each run, termination information is entered in the master log. The entry format is:

Table 9-5. Run Termination Log Entry

0	*entry-type (5)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1 2	*account-identifier*					
3 4	*project-id*					
5	*run-initiation-time (TDATE$ format)*					
6	*run-termination-time (TDATE$ format)*					
7	*cards-in*			*cards-out*		
8	*priority*	*page-count-using-standard-page-size*				
9	*estimated-run-time-in-SUPs*					
10	*actual-run-time-in-SUPs*					
11	*core-block-SUPs*					

Table 9-5. Run Termination Log Entry (continued)

12 ≈ 19	*Words 12 through 19 contain the parameters Tracks*(SUPs + voluntary-delay-time) for each of the fixed mass storage device groups. Tracks applies to temporary files and expansion of cataloged files assigned to the run.*	
20 21	*user-id*	
22	*total-number-of-sectors allocated/released*	*total-number-of allocation/release-calls*
23	*granule-table-r/w*	*total-number-of directory-control-calls*
24	*total-number-of-PRINT$-pages*	*PRINT$-pages-since-last-BRKPT*
25	*date-and-time-of-log-entry*	
26	*reserved*	
27	*run-id*	

Word 23

granule-table-r/w A count of the read/write operations used to maintain granule tables.

total-directory control-calls The total count of references to the Executive cataloging routines to support all cataloged files in the run.

9.6. I/O Error Log Entry

A record of all I/O errors is kept in the master log. A count of valid references to a unit is maintained in main storage and, when an error occurs, this information along with the error information is placed in the master log. The reference count is cleared to zero at that time. Note that after a predetermined number of retries (number of retries is device dependent), all of which fail, the operator is notified and operator intervention is required. The entry formats are shown and described in Tables 9-6, 9-7, and 9-8.

Table 9–6. Channel Program Error/Unsolicited Interrupt Log Entry (1100/80, 60)

0	*entry–type (6)*	*nbr–wrds in–entry*	*system indicator*	reserved	reserved	nbr–of–log entries–in 224–wrd–blk
1	error–type	sense byte–count	unit–equipment–type		control–unit equipment–type	
2	unit–device–name					
3	control–unit–device–name					
4	user function code	handler function code	nbr–of–lost–log–entries		retry–count (neg–if–retry–failed)	
5	I/O–path–used			system type	error–message index	
6	number–of–references–since–last–try					
7	channel–command–word–0					
8	channel–command–word–1					
9	channel–status–word–0					
10	channel–status–word–1					
11	channel–status–word–2					
12	up–to–six–words–of–sense–bytes					
17						
18	up–to–six–words–of–device–dependent–information					
23						
24	reel–id–if–tape, pack–id–if–disk					
25	date–and–time–of–log–entry					
26	absolute address of first command CCW in this segment					
27	run–id–name					

Table 9-7. Channel Program Error/Unsolicited Interrupt (1100/60)

0	entry-type (6)	nbr-wrds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1	error-type	sense byte-count	unit-equipment-type		control-unit equipment-type	
2	device-name					
3	control-unit-name					
4	user function code	handler function code	nbr-of-lost log-entries		retry-count (neg-if-retry-failed)	
5	I/O-path-used			system type	error-message-index	
6	number-of-references-since-last-try					
7	access-control-word					
8	chain-pointer-word					
9	EI-status-word					
10	MSA-auxiliary-status					
11	logical control unit number	logical-MSA number	unused			
12 ≈ 17	up-to-six-words-of-sense-bytes					
18 ≈ 23	up-to-six-words-of-device-dependent-information					
24	reel-id-if-tape, pack-id-if-disk					
25	date-and-time-of-log-entry					
26	absolute address of first command ACW in this segment					
27	run-id-name					

Word 1

error-type	0	channel program error
	1	unsolicited interrupt
	2	nonsense interrupt

Word 5

system-type	2	1100/60, 1100/80

Words 18–23

If the associated device is a tape, then the data in Words 18 through 23 have the following format:

18	*nbr-of-files-extended*	*channel program condition code*	*nbr-of-blocks-extended*
19	*reserved*		
20	*command-bytes (2-words)*		
21	*or 2 EF words*		
22	*reserved*		
23	*reserved*		

If the associated device type is mass storage, then the data in Words 18 through 23 have the following format:

18	*mass-storage error-code*	*retry-count-on error-type*	*word/record from-USTWDR*
19	*device relative word address*		
20	*command-bytes (2-words)*		
21	*or 2 EF words*		
22	*error offset from label*	*label (continued to next word)*	
23	*reserved*		

If the associated device is a symbiont, then the data in Words 18 through 23 have the following format:

18	*run-id (print only)*	
19		*symbiont equipment index*
20	*command-bytes (2-words)*	
21	*or 2 EF words*	
22	*reserved*	
23	*reserved*	

Table 9-8. Nonsense Interrupt Log Buffer Format

0	*entry-type (6)*	*nbr-of-wrds in-the log-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*error-type 2*	*nbr-of interrupts in-the log-entry*	*nbr-of interrupts thrown away*		*nbr-of-interrupts-in-stack*	
2	*reserved*					
3	*nonsense-interrupt-save-area(20-words)*					
≈	*(up-to-five-4-word-entries)*					≈
22						
23	*reserved*					
24	*reserved*					
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

When an I/O error occurs on a SSP diskette drive (1100/60 only), an entry is made in the master log as shown:

Table 9–9. SSP Diskette Type 6 Log Entry

0	*entry-type (6)*	*nbr-of-wrds in-the log-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*error-type = 3*	*SSP subtype = 1*	*SSP number*			*disk I/O status*
2						*disk device address*
3						*I/O command that failed*
4						*buffer addr (low byte)*
5						*buffer addr (high byte)*
6						*track number*
7						*sector number*
010						*sector count*
011						*retries attempted*
012		*V*	*O*	*L*		*U*
013		*M*	*E*	*I*		*D*
014						
≈	*unused*					
036						
037	*date-and-time-of-log-entry*					
040						
041						

NOTE: Words 012 and 013 (VOLUMEID) U, D are bits 7-0, L, I bits n-n, O, E ...,

When a fault is detected on the SSP maintenance interface (MIA) (1100/60 only), the error information is placed in the system log:

Table 9-10. SSP Maintenance Interface (MIA) Type 6 Log Entry

0	entry-type (6)	nbr-of-words in-the log-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1	error type = 3	SSP subtype = 2	SSP number			sense byte
2						support controller byte 0
3						support controller byte 1
4						support controller byte 2
						dev status
						dev adrs
						timeout analysis code (see below)
036						
037	date-and-time-of-log-entry					
040						
041						

Timeout
Analysis Codes:
0 = cause unknown
1 = MIA or SC
2 = IOU waiting for ack1
3 = IOU waiting for ack2
4 = MSU was stop on error
5 = MSU had invalidate request up
6 = MSU clock was stopped
7 = CPU clock stopped
8 = address disable up
9 = micro code branching
10 = CPU waiting for must acknowledge
11 = CPU waiting for IOU initiate acknowledge

9.7. Console Log Entries

Each console message is placed in the master log. At run termination, every message pertaining to the run is printed at the end of the program listing. The entry format is:

Table 9-11. Console Log Entries

0	*entry-type (7)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*spaces*				*msg-nbr*	*space*
2	*message (23-word-maximum)*					
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

msg-nbr - A digit if this is the output portion of a type-and-read.

9.8. Checkpoint Log Entry

When a checkpoint is used in a run, an entry is made in the master log with pertinent information concerning the checkpointed run. The entry format is:

Table 9-12. Checkpoint Log Entry

0	*entry-type (8)*	*nbr-of-wds in-entry*	*system indicator*	*unused*	*unused*	*nbr-of-log entries-in 224-wrd-blk*
1	*message (24-word maximum)*					
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

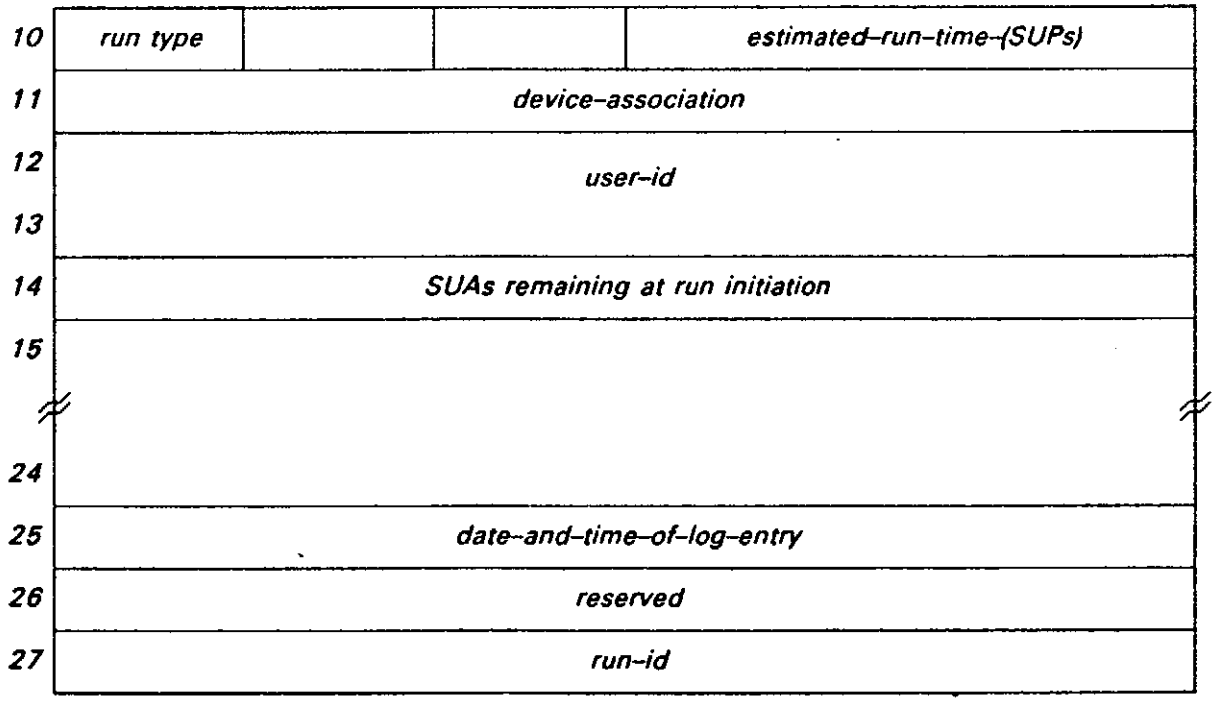
9.9. Run Initiation Log Entry

When a run is opened, an entry is made in the master log with the pertinent information concerning the run. The entry format is:.

Table 9-13. Run Initiation Log Entry

0	*entry-type (9)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*options*	*priority*	*start-time (in minutes)*		*deadline-time (in minutes)*	
2	*estimated-pages-out*			*estimated-card-out*		
3	*run-id (new or generated)*					
4	*run-id (old or original)*					
5	*project-id*					
6						
7	*account-number*					
8						
9	*sequence-id*					

Table 9-13. Run Initiation Log Entry (continued)



Word 1

options The possible values of options specified on the @RUN control statement are:

010	T
004	P
002	C
001	S

Word 9

sequence-id If the run is from a batch input device, this parameter is the run-id of the preceding run from the same device. If the run was scheduled via @START, this parameter is 0.

Word 10

run type The possible run types are:

004	demand
005	deadline batch
006	batch

Word 11

device-association This parameter is the Fieldata name of the device that read the run, or, in the case of a run scheduled via @START, this parameter is 0.

This entry is always the first one in the first block of a series of contiguous blocks in the master log file for a given run. The run-id field (new or generated) contains the same identity as Word 27 of each entry for the subject run.

9.10. Console Replies Log Entry

Replies to console type and read messages are placed in the master log. The replies as well as the type and read messages are printed at the end of the program listing. The entry format is:

Table 9-14. Console Replies Log Entry

0	*entry-type (10)*	*nbr-of-words in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*spaces*			*space*	*msg-nbr*	*space*
2 21	*message (20-word-maximum)*					
22	*site-id*					
23-24	*user-id-of-issuing-run*					
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

9.11. Log Keyin Entry

Each time a LG keyin is input from the operators console, the message will be recorded in a log entry. The entry format is:

Table 9-15. Log Keyin Entry

0	*entry-type (11)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*message (9-word-maximum)*					

Table 9-15. Log Keyin Entry (continued)

24	
25	*date-and-time-of-log-entry*
26	*reserved*
27	*EXEC 8* *(will be EXEC 8 if no run-id is specified on the LG keyin)*

9.12. Unsolicited Keyin Log Entry

The entry format for unsolicited keyin log entries is:

Table 9-16. Unsolicited Keyin Log Entry

0	*entry-type (12)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*spaces*			*actual-keyin-in-Fieldata-format (left-justified)*		
2	*message (20-word-maximum)*					
≈						
21						
22	*site-id*					
23-24	*user-id-of-issuing-run*					
25	date-and-time-of-log-entry					
26	reserved					
27	EXEC 8					

9.13. Tape Labeling Log Entry

When the tape labeling feature of the Executive is used, log entries are made in the master log. These entries contain pertinent information concerning allocation and release of tape reels, and errors encountered during tape labeling. The entry format is:

Table 9-17. Tape Labeling Log Entry

0	*entry-type (13)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*message (24-word-maximum)*					
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

9.14. Symbiont End of Processing Log Entry

When a symbiont finishes processing an input or output file, an entry is made in the EXEC chain on the master log with pertinent information concerning the processed file. If the type 15 log entry resulted from a started (@START) run, there will be no corresponding type 14 log entry. The entry format is:

Table 9-18. Symbiont End of Processing Log Entry

0	*entry-type (14)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*equipment-code*		*file-type*	*total page/card count*		
2	*symbiont-name*					
3	*line-count-or-cards-in/out*					
4	*run-id-of-associated-run*					

Table 9-18. Symbiont End of Processing Log Entry (continued)

5	*account-number*			
6				
7	*user-id if LOGONS > 0 and RSI device*			
8				
9	*output filename*			
10				
11	*output filename qualifier*			
12				
13	*pages/cards reprinted*	*pages/cards skipped*		
14	*reserved*	*reserved*	*reserved*	*termination type*
15				
≈	*reserved*			≈
24				
25	*date and time of log entry*			
26	*reserved*			
27	*EXEC 8*			

Word 1

equipment-code equipment code of symbiont device from SYMACT if not an onsite device, or from USTSEI in Unit Status table if onsite device

Log type 14 and 15 equipment code entries:

Onsite Devices (octal):

03 0716 MSA/Byte card reader
027 0604 MSA/Byte card punch
055 0768 MSA or Byte channel printer
056 0770 MSA or Byte channel printer
057 0776 MSA or Byte channel printer
076 C/SP device

Remote Devices (octal):

02 U100/DCT-1000
03 U300
05 TTY
06 DCT-500

Remote Batch (010–017):

010 Remote 1004/9000 (REM1)
011 DCT 2000
012–016 not used
017 Full Duplex Remote (FDR)

RSI Table Type Use (020–027):

020 RSI Primary Input
021 RSI Additional Input
022 RSI Output
023 RSI Demand with Generic Siteid
024 RSI Console (@@CONS callers)

030–037 Not Used

file type has value of:

01 input cards
02 output cards
03 output pages

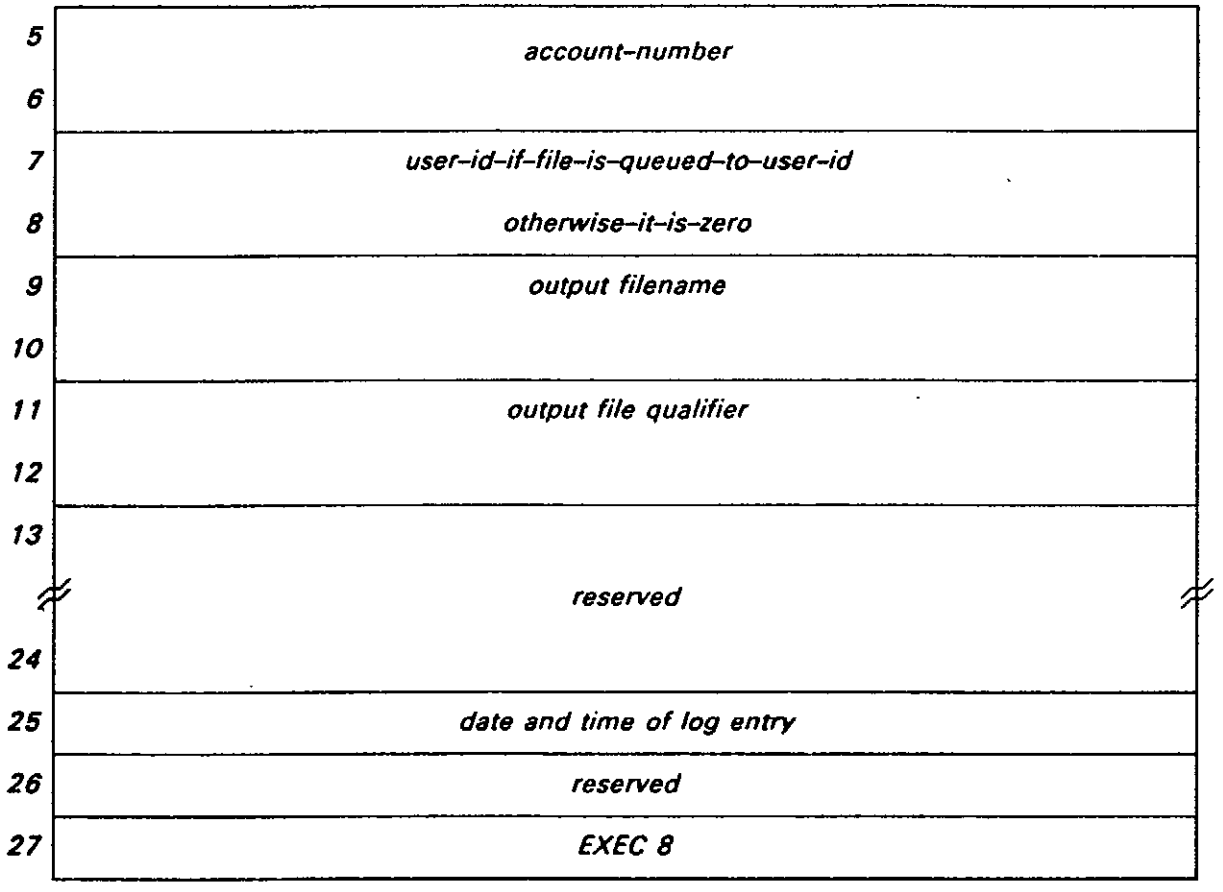
9.15. Symbiont Start of Processing Log Entry or @START Log Entry

When a symbiont begins processing an input or output file, or an @START is done, an entry is made in the EXEC chain on the master log with pertinent information regarding the file to be processed or the run to be started (@START). The entry format is:

Table 9–19. Symbiont Start of Processing Log Entry

0	*entry–type (15)*	*nbr–of–wds in–entry*	*system indicator*	*reserved*	*reserved*	*nbr–of–log entries–in 224–wrd–blk*
1	*equipment–code*		*file–type*			
2	*symbiont–name*					
3						
4	*run–id–of–associated–run*					

Table 9-19. Symbiont Start of Processing Log Entry (continued)



Word 1

equipment-code equipment code of symbiont device (see log type 14). For started (@START) runs, zero is inserted as equipment code.

file-type . has value of:

01 input cards
02 output cards
03 output pages

9.16. Program Initiation Log Entry

When a program is initiated, a log entry is made to record pertinent values that are cumulative during the run. The entry format is:

Table 9-20. Program Initiation Log Entry

0	*entry-type (16)*	*nbr-of-words in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1 2	*program name*					
3 4	*version-name*					
5	*program-initiation-date/time*					
6	*CPU-time*					
7	*quantum-timer-value-in-100-nanoseconds-intervals (1100/80) or quantum-timer-value-in-116-nanoseconds-intervals (1100/60)*					
8	*ER + control-statement-charge (200 microseconds)*					
9	*I/O-transfer-cost-group-1-devices*					
≈	≈					≈
18	*I/O-transfer-cost-group-10-devices*					
19	*SUPs (200 microseconds)*					
20	*CBSUPs (core-block-SUPs)*					
21	*voluntary-delay-time (200 microseconds)*					
22	*time-in-real-time-mode (200 microseconds)*					
23	*unused*					
24	*condition-word*					
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

9.17. Run Termination Supplement Log Entry

The entry format for run termination supplement log entries is:

Table 9-21. Run Termination Supplement Log Entry

0	*entry-type (17)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*CPU-time*					
2	*quantum-timer-value-in-100-nanoseconds-intervals (1100/80) or quantum-timer-value-in-116-nanoseconds-intervals (1100/60)*					
3	*ER-and-control-card-charge (200 microseconds)*					
4	*I/O-transfer-cost - group-1-devices*					
≈						
13	*I/O-transfer-cost - group-10-devices*					
14	*voluntary-delay-time (200 microseconds)*					
15	*1100/80 CPU-time (200-microsecond-increments) 1100/60 CPU-time (116-nanosecond-increments)*					
16	*SUAs-used*					
17						
≈						
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

9.18. Recovery Close-Out Log Entry

When a system recovery is performed before a run has gone through run termination accounting, the sequence of log entries for that run may not contain the type 17 log entry which effectively closes out the set of log entries for that run. In this case, the following log entry is added to the set of log entries at the time of system recovery. This log entry effectively closes an open set of log entries. The entry format is:

Table 9-22. Recovery Close-Out Log Entry

0	*entry-type (18)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	"SYSTEM HAS ABORTED. LOG RECOVERED AT: *hhmmss*"					
6						
7						
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*run-id*					

9.19. Cooperative Accounting

When a PRINT$ or PUNCH$ file is breakpointed (@BRKPT), or when an alternate print or punch file is closed, a log entry will be inserted into the log. The entry format is:

Table 9-23. Cooperative Accounting

0	*entry-type (19)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*PRLEN*		*PRCUR*		*PRTOP*	*PRBOT*
2	*pages/cards*					

Table 9-23. Cooperative Accounting (continued)

3	*file-type-indicator*	*pages/cards since last BRKPT*
4	*filename*	
5		
6		
//		
24		
25	*date-and-time-of-log-entry*	
26	*reserved*	
27	*run-id*	

Word 1

PRLEN	Current length of page less bottom margin
PRCUR	Current position on page
PRTOP	Current top margin
PRBOT	Current bottom margin

Word 3

File type indicator	Type 2 for punch file, type 3 for print file.

9.20. Facility Usage Summary Log Entry

The facility usage summary log entry is added to the user chain after the type 17 if any tape, removable disk, or symbiont units were assigned by the run. Its format is:

Table 9-24. Facility Usage Summary Log Entry

0	*entry-type (20)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*

Table 9-24. Facility Usage Summary Log Entry (continued)

1	*tape-unit-SUPs-in-seconds (Group 1)*		
≈	≈	≈	≈
	tape-unit-SUPs-in-seconds (Group n)		
	disk-unit-SUPs-in-seconds (Group 1)		
≈	≈	≈	≈
	disk-unit-SUPs-in-seconds (Group n)		
	symbiont-unit-SUPs-in-seconds (Group 1)		
≈	≈	≈	≈
23	*symbiont-unit-SUPs-in-seconds (Group n)*		
24	*nbr-of-tape-group-entries*	*nbr-of-disk-group-entries*	*nbr-of-symbiont group-entries*
25	*date-and-time-of-log-entry*		
26	*reserved*		
27	*run-id*		

9.21. Symbiont Close-Out Log Entry

When the last output file for a run has been processed, an entry is made in the EXEC log chain. The entry format is:

Table 9-25. Symbiont Close-Out Log Entry

0	*entry-type (21)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*run-id-of-associated-run*					
2						
≈	≈	≈	≈	≈	≈	≈
24						

Table 9-25. Symbiont Close-Out Log Entry (continued)

25	*date-and-time-of-log-entry*
26	*reserved*
27	*EXEC 8*

9.22. EXEC Segment Validation Log Entry

When an error is encountered while loading an EXEC segment, a log entry is made and subsequently placed in the Master Log. The entry format is:

Table 9-26. EXEC Segment Validation Log Entry

0	*entry-type (22)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*mass-storage-address-of-segment*					
2	*keyword-or-checksum-received*					
3	*'$FUNC$'-or-checksum-expected*					
4	*ANS = (A,B,or G)*		*number-of-attempts-to-load-segment*			
5	*segment-name*					
6						
//						
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*EXEC 8*					

A type 22 log entry is also generated when an error is encountered reading a mass storage master bit table. In this case the entry format is:

Table 9-27. Mass Storage Master Bit Table Error

0	*entry-type (22)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*mass-storage-address-of-MBT*					
2	*checksum-expected*					
3	*checksum-received*					
4	*reserved*					
5	*'MBTERR'*					
6						
≈						≈
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*EXEC 8*					

9.23. Software Instrumentation Package Log Entry

The type 23 log entry has been reserved for use by Software Instrumentation Package (SIP). Entries are variable in length and always constitute one physical block of the log file on tape.

9.24. Software Detected Error Log Entry

Type 24 has been reserved for software detected error log entries. (See Tables 9-28 through 9-34.) Several different entries (and formats) have been defined and are described below.

The log entry shown in below is created by FREL whenever a track of mass storage to be released by FREL is not allocated to the file.

Table 9-28. Software Detected Error Log Entry (Subcode 0001)

0	entry-type (24)	nbr-of-wds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					subcode (0001)	
2	'FREL'					
3	qualifier					
4						
5	filename					
6						
7	user-run-id					
8	assign-options-from-file-item					
9	initial-reserve			maximum-granules		
10	highest-track-referenced			highest-granule-assigned		
11	count of granules released			rel-cycle number	absolute cycle-number	
12	pack-id					
13	caller					
14		MBT flag	FRESC call	unit-status-table		
15	equipment-type		0=FASTRAND 1 = drum		removable if nonzero	00 = TRK 40 = POS
16	logical device name					
17	IDL-location					
18	number-of-tracks-released				nbr-of-positions now-available	
19	current track-nbr					
20	starting-track-nbr-being-released					
21	original-request-limits					
22	bit-pattern-for-position-where					

Table 9-28. Software Detected Error Log Entry (Subcode 0001) (continued)

23	release-cannot-be-made
24	request-limits-at-time-of-malfunction
25	date-and-time-of-log-entry
26	reserved
27	EXEC 8

Table 9-29. Software Detected Error Log Entry (Subcode 0003)

0	entry-type (24)	nbr-of-wds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					subcode (0003)	
2	'UPROAR' or 'UPREG'					
3	pack-id					
4	qualifier					
5						
6	filename					
7						
8	project-id-on-the-pack					
9						
10	account-on-the-pack					
11						
12	time-of-last-reference					
13	time-of-cataloging					
14	f-cycle		0	nbr-times-assigned		
15	equipment-type					

Table 9-29. Software Detected Error Log Entry (Subcode 0003) (continued)

16	*project-in-the-MFD*		
17			
18	*account-in-the-MFD-or-0*		
19			
20	*time-of-last-reference-or-0*		
21	*time-of-cataloging-or-0*		
22	*equipment-type*		*nbr-of-times-assigned-or-0*
23			*error-address*
24			
25	*date-and-time-of-log-entry*		
26	*reserved*		
27	*EXEC 8*		

When a removable disk pack is being registered and a directory track is lost, the following log entry is created. The log entry format is shown in Table 9-30.

Table 9-30. Software Detected Error Log Entry (Subcode 0004)

0	*entry-type (24)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1					*subcode (0004)*	
2	*'UPROAR'*					
3	*removable-disk-pack-id*					
4	*initial-directory-track-address*					
5	*status*	*I/O function*				
6	*access-word*					
7	*disk/drum-address*					

Table 9-30. Software Detected Error Log Entry (Subcode 0004) (continued)

8		error-address
9	first-word-initial-DAS	
10	last-word-initial-DAS	
11	current-word-current-DAS	
12		
//		//
24		
25	date-and-time-of-log-entry	
26	reserved	
27	EXEC 8	

When a removable disk pack cannot be registered due to some unrecoverable error, the following log entry is created. The log format is shown in Table 9-31.

Table 9-31. Software Detected Error Log Entry (Subcode 0005)

0	entry-type (24)	nbr-of-wds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					subcode (0005)	
2	'UPROAR'					
3	pack-id					
4	initial-directory-address					
5	status	I/O function				
6	access-word					
7	disk/drum-address					
8				error-address		
9	directory-lock-word (FIFPTS)					

Table 9-31. Software Detected Error Log Entry (Subcode 0005) (continued)

10	FIPTLK		FATULK
11			
≈			
24			
25	date-and-time-of-log-entry		
26	reserved		
27	EXEC 8		

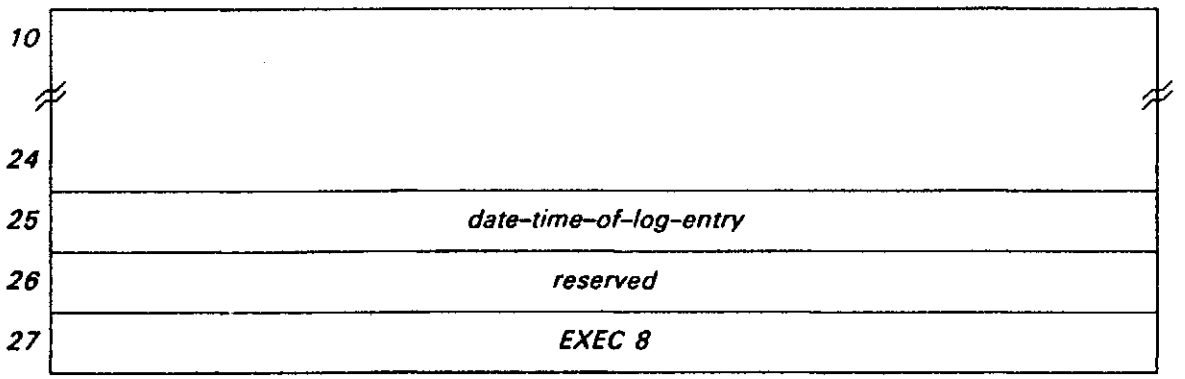
The log entry shown in Table 9-32 is created when a C/SP illegal function or communications error is encountered.

The purpose of this log entry is to record information about fatal system errors, nonfatal system errors, and system boots.

Table 9-32. Fatal, Non-Fatal System Errors and System Boots Log Entry

0	entry-type (24)	nbr-of-wds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					subcode (0006)	
2	XERCODE			XERSCODE		
3	XERCPU	XERTYP		XERPADD		
4	XERDATE			XERTIME		
5	XERSES			XERSEQ		
6	XERJKS			XERSEG		
7		XERBTP	XERBTC	XERNSES		
8						
9	XERDEP					

Table 9-32. Fatal, Non-Fatal System Errors and System Boots Log Entry (continued)



Word 3

The value of the field XERTYP indicates the contents of the log entry:

XERTYP	Log Entry Contents
0	Fatal system error information and boot information
1	Non-fatal system error information
2	Only boot information

The fields of the log entry have the following meaning for each value of XERTYP:

Field	XERTYP 0	XERTYP 1	XERTYP 2	Definition
XERCODE	X	X		System error code
XERSCODE	X	X		System error subcode
XERCPU	X	X		CPU number of reporting CPU
XERPADD	X	X		Address from which error was reported
XERTIME	X			Time in seconds since midnight of fatal error
XERDATE	X			Date of fatal error mm dd yy octal
XERSES	X	X		Session in which error occurred
XERSEQ	X	X		Sequence number of latest nonfatal error
XERSEG	X	X		Segment index of a function that reports an error
XERDEP	X			Contains information that is dependent upon the type of fatal system error
XERBTP	X		X	Master bit settings for boot type:
XERJKS	X		X	Jump key settings on reboot, master bit format
XERBTC	X		X	Boot CPU number
XERNSES	X		X	Number of the session initiated by the boot

Bit	Value	Boot type
0	0	Tape boot
	1	Drum/disk boot
1	0	Initial boot
	1	Recovery boot
2	0	Manual boot
	1	Auto-boot
3	0	No panic dump
	1	Panic dump taken
4	0	Software-initiated recovery
	1	Hardware-initiated recovery

The communications software error log entry format is shown in Table 9-33.

Table 9-33. Software-Detected Error Log Entry (Subcode 0007)

0	*entry-type (24)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wd-blk*
1					*subcode (0007)*	
2	*word-1-of-status-information*					
3	*word-2-of-status-information*					
4	*word-3-of-status-information*					
5						
≈						
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*EXEC 8*					

The C/SP error log entry has the format shown in Table 9-34 and described below.

Table 9-34. Software Detected Error Log Entry (Subcode 0073)

0	*entry-type (24)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1			*LGTYPE*	*C/SP-nbr*	*subcode (0073)*	
2	*QCCI0*					
3	*QCCI1*					
4						
≈	*QCI0 Data ...*					
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					

Table 9-34. Software Detected Error Log Entry (Subcode 0073) (continued)

27	EXEC 8

Word 1

LGTYPE
- 0 C/SP initiated log entry
- 1 Illegal function from the C/SP
- 2 Communications Error

9.25. Checkpoint Initiation Log Entry

Whenever a CKPT operation is initiated, a log entry is created and subsequently placed in the Master Log. The entry format is:

Table 9-35. Checkpoint Initiation Log Entry

0	entry-type (25)	nbr-of-wds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1	primary-storage-reference-count					
2	primary-storage-reference-count					
3	extended-storage-reference-count					
4	extended-storage-reference-count					
5	ER+control-card-charge					
6	I/O-SUPs					
≈						
15						
16	voluntary-delay-time					
17	CPU-time					
18						
≈						
24						
25	date-and-time-of-entry					
26	reserved					

Table 9-35. Checkpoint Initiation Log Entry (continued)

27	run-id

9.26. Restart Initiation Log Entry

Whenever a restart (RSTRT) operation is initiated a log entry is made and subsequently inserted into the Master Log. The entry format is:

Table 9-36. Restart Initiation Log Entry

0	entry-type (26)	nbr-of-wds in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1	primary-storage-reference-count					
2	primary-storage-reference-count					
3	extended-storage-reference-count					
4	extended-storage-reference-count					
5	ER+control-card-charge					
6	I/O-SUPs					
≈						
15						
16	voluntary-delay-time					
17	CPU-time					
18						
≈						
24						
25	date-and-time-of-entry					
26	reserved					
27	run-id					

9.27. Hardware Fault Log Entry

Tables 9–37 through 9–42 are the log entries for 1100/60, and 1100/80 systems hardware interrupts where recovery is attempted.

Whenever one of these interrupts occurs, an entry will be made in the master log. However, if the error is fatal, a log entry will not be made.

Table 9–37. I/O Fault Log Entry (Non–1100/60/80)

0	*entry–type (27)*	*nbr–of–words in–entry*	*system indicator*	*reserved*	*reserved*	*nbr–of–log entries–in 224–wd–blk*
1	*reserved*	*handler function code*	*logical–MSA number*	*path–used–by–this–I/O*		
2	*unit–status*	*reserved*	*log–info lost–flg*	*reserved*	*fault–type*	
3	*access–control–word*					
4	*channel–chain–pointer–word*					
5	*EI–status–word*					
6 ≈ 10	*reserved*					
11	*CPU/CAU–name*					
12	*IOU/IOAU–name*					
13	*control–unit–name*					
14	*unit–name*					
15	*reserved*	*CU number*	*unit–equipment type–index*	*control–unit equip–type–index*		
16	*references–since–last–error*					
17	*fault–status–word*					
18	*retry–count (neg–if–retry–failed)*	*reserved*				
19 ≈	*reserved*					

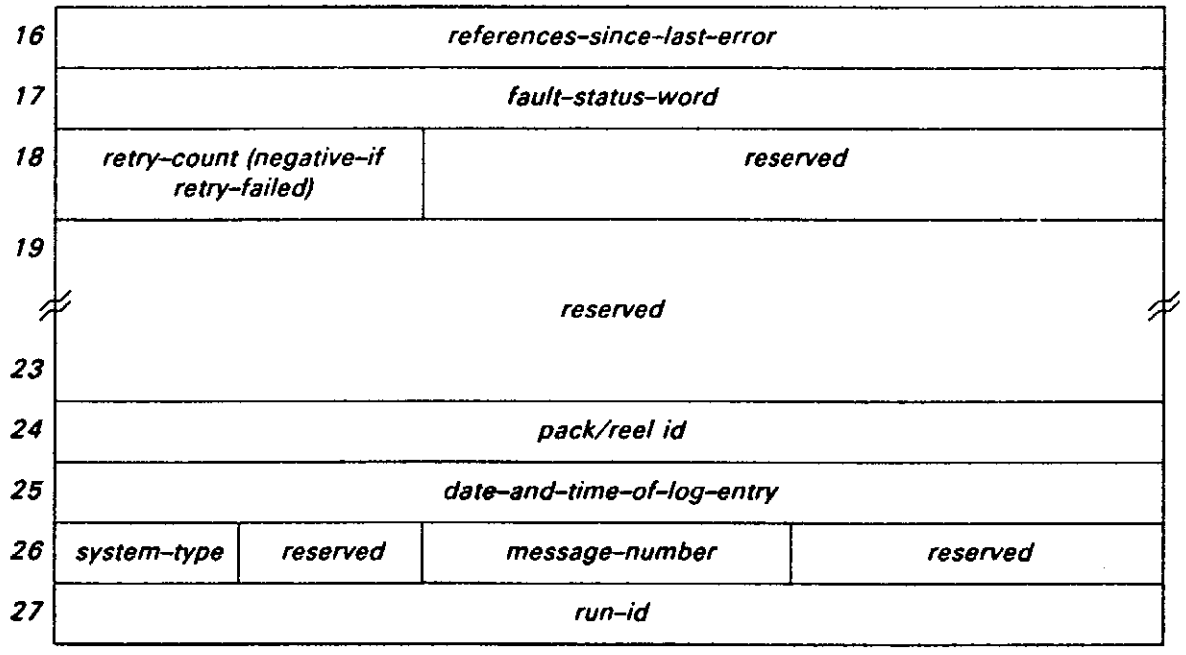
Table 9–37. I/O Fault Log Entry (Non–1100/60/80) (continued)

23				
24	reel/pack id			
25	date–and–time–of–log–entry			
26	system–type	reserved	message–number	reserved
27	run–id			

Table 9–38. I/O Fault Log Entry (1100/60/80)

0	entry–type (27)	nbr–of–words in–entry	system indicator	reserved	reserved	nbr–of–log entries–in 224–wd–blk
1	reserved	handler function code	reserved	path–used–by–this–I/O		
2	unit–status	instruction type	log–info lost–flg	nonzero condition code	fault–type	
3	interrupt–address–word					
4	channel–status–word–0					
5	channel–status–word–1					
6	channel–status–word–2					
7	channel–command–word–0					
8	channel–command–word–1					
9	reserved					
10	reserved					
11	CPU–name					
12	IOU–name					
13	control–unit–name					
14	unit–name					
15	reserved	CU number	unit–equipment type–index	control–unit equip–type–index		

Table 9-38. I/O Fault Log Entry (1100/60/80) (continued)



where:

fault-type	Type	Definition
	001	
	004	
	005	
	006	
	010	
	011	
	012	
	013	
	017	first log buffer–1
	020	nonsense interrupt
	021	nonzero condition code from 'CIOIER'
	022	nonzero condition code from 'SRLSUB'
	023	sense failure detected by 'SRLINT'
	024	nonzero condition code from 'SRLUNL'
	025	machine check
	026	interrupt discarded as 'USTATE' greater than zero
	027	subchannel status errors
	030	timeout
	031	nonzero condition code
	032	unsolicited interrupt
	033	I/O interrupt error

instruction type

000 – SIOF
001 – TIO
002 – TSC
003 – HDV
004 – HCH
007 – LCBR

010 - LTCW

Table 9-39. Processor Fault Log Entry

0	*entry-type 27*	*nbr-of-wrds in-the log-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*reserved*					
2					*fault-type=040*	
3	*flg 1*	*processor fault-type*	*CPU-nbr*			
4	*CPU-name*					
5						
6						
7	*status 1*					
8	*status 2 for MDA or MDB failure*					
9						
≈						
16						
17						
≈	*contents of SCRLOG entry*					
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*EXEC 8*					

Word 3:

processor fault-type	1100/80	1100/80A	Description
	024	026	Diagnostic MDA Failure
	025	027	Diagnostic MDB Failure
	026	030	Prom Parity Successful
	027	031	Prom Parity Unsuccessful
	030	032	Power Loss/Restoration Interrupt

flg - see values following Table 9-41.

Table 9-40. Processor Fault Log Entry 1100/60 (Internal Check)

0	entry-type 27	nbr-of-wrds in-the log-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1	reserved					
2					fault type = 040	
3	flg 1	fault-type	CPU-nbr	time-of-error	day-of-the-year	
4	CPU-name					
5						
6	SIU-name					
7	status-word					
8	BDIs-of-interrupted-environment 2-words					
9						
10	D-bits-of-interrupted-environment					
11	P-address-of-interrupt					
12	interrupted-run-id					
13	hardware-retry-time-value-when-fault-occurs					
14						
≈						
16						
17						
≈	contents of SCRLOG entry					
24						
25	date-and-time-of-log-entry					
26	reserved					
27	EXEC 8					

Word 3:

fault type	40	Immediate Internal Check 1
	41	Immediate Internal Check 2
	42	Delayed Internal Check 1
	43	Delayed Internal Check 2

flg – see values following Table 9–41

Table 9–41. Storage Fault Log Entry 1100/80/80A/60

0	*entry–type 27*	nbr–of–wrds in–the log–entry	system indicator	reserved	reserved	nbr–of–log entries–in 224–wrd–blk
1	*reserved*					
2	*MSU TU position number*	*nbr–of failing retries*	*flg 1*	*number of recurring SBECs*	*fault–type=041*	
3	*flg*	*storage fault–type*	*CPU–nbr*	*time–of–error*	*day–of–the–year*	
4	*MSU–name*					
5	*MSU–relative–address*					
6	*SIU–name*					
7	*status–word*					
8	*BDIs–of–interrupted–environment*					
9						
10	*D–bits–of–interrupted–environment*					
11	*P–address–of–interrupt*					
12	*interrupted–run–id*					
13	*unused–(1100/60/80/80A)*			*hardware–retry–timer–(1100/60)*		
14	*initial nbr blks degraded (80A/60)*			*current nbr blks degraded (80A/60)*		
15	*unused*					
16	*unused*					
17	*contents–of–SCRLOG*					
≈						
24						

Table 9-41. Storage Fault Log Entry 1100/80/80A/60 (continued)

25	*date-and-time-of-log-entry*
26	*reserved*
27	*EXEC 8*

Word 2

fault type The storage fault is characterized by the presence of code 041.

flg 1

001	true retry
002	simulated retry
003	no retry permitted
010	fault occurred during bootstrap (1100/80A and 1100/60)

Word 3

flg

001	log flag
002	count against threshold
004	error caused reboot
010	software retry failure (1100/80 and 1100/80A) hardware retry failure (1100/60)
020	EXEC software retry successful (1100/80 and 1100/80A) SSP software retry successful (1100/60)
040	processing complete

storage fault type (1100/80)

001	ISC - Address or Control Parity Mode
002	ISC - Control Parity Error
003	ISC - Address Parity Error
004	ISC - Write Data Check
005	ISC - Read Data Check
006	ISC - Address Not Available
007	Guard Mode IPT CPU SIU I/Face Offline
010	DSC - Tag Parity Error
011	DSC - Age Check
012	DSC - LRU Check
013	DSC - Storage Offline
014	DSC - Write Data
015	DSC - Write Address Parity Error
016	DSC - Read Address Parity Error
017	DSC - Read Data
020	DSC - Storage Correctable (Single Bit)
021	DSC - Storage Uncorrectable (Multiple Bit)
022	ISC - Illegal Status
023	DSC - Illegal Status

storage fault type (1100/80A)

001	ISC - Address or Control Parity Mode
002	ISC - Write Control Parity Check
003	ISC - Write Data Parity Check
004	ISC - Address Not Available
005	ISC - Special ECC Syndrome
006	ISC - Read Parity Data Check
007	ISC - No Meaningful Indication

	010	Guard Mode IPT CPU SIU I/Face Offline
	011	DSC – Invalidate Address Check
	012	DSC – Data Check Upper
	013	DSC – Data Check Lower
	014	DSC – Tag Buffer Check
	015	DSC – Age Check
	016	DSC – Maintenance Read Miss
	017	DSC – Write Data Check
	020	DSC – Address Check
	021	DSC – Write Control Check
	022	ISC – Multiple Uncorrectable Error
	023	DSC – Single Bit Error
	024	DSC – Illegal
	025	DSC – Illegal
storage fault type (1100/60)	1	ISC – Multiple Uncorrectable Error
	2	ISC – Address Not Available
	3	ISC – Special Error Code
	4	ISC – Read Data Parity Check
	5	ISC – Write Data Parity Check
	6	ISC – Control Parity Check
	7	ISC – Address Parity Check
	010	DSC – Data Check Upper
	011	DSC – Data Check Lower
	012	DSC – Tag Buffer Check
	013	DSC – Age Parity Check
	014	DSC – Invalidate Address Check
	015	DSC – Maintenance Read Miss
	016	DSC – Address Not Available
	017	DSC – Data Parity Check, Write Request
	020	DSC – Data Parity Check, Read Request
	021	DSC – Address Data Check, Write Request
	022	DSC – Address Data Check, Read Request
	023	DSC – Write Control Check, Write Request
	024	DSC – Write Control Check, Read Request
	025	DSC – Multiple Uncorrectable Error
	026	DSC – Single Bit Error
	027	DSC – Multiple Uncorrectable Error
	030	DSC – Address Not Available
	031	DSC – Special Error Code
	032	DSC – Read Data Parity Check
	033	DSC – Write Data Parity Check
	034	DSC – Control Parity Check
	035	DSC – Address Parity Check
	036	ISC – Illegal Status
	037	DSC – Illegal Status
	044	No-Error-Status

Table 9-42. Down Component Log 1100/60/80/80A

0	*entry-type 27*	*nbr-of-wrds in-the log-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*reserved*					
2					*fault-type=041*	
3	*flg*	*storage fault-type*	*CPU-nbr*	*time-of-error*	*day-of-the-year*	
4	*component-name*					
5						
6						
7	*TU-position-number*					
8						
≈						
11						
12	*interrupted-run-id*					
13						
≈						
16						
17						
≈	*contents-of-down-component-SCRLOG*					
24						
25	*date-and-time-of-log-entry*					
26	*reserved*					
27	*EXEC 8*					

flg - see values following Table 9-41

storage fault type 1100/80	031	Down CPU
	032	Down SIU Segment
	033	Down MSU Bank
	034	Down 64-Word Block
	035	Down Then Up 64-Word Block
	036	Down CPU-SIU I/Face

	037	Down Cluster
	040	Down Chip
	041	Down SIU Half
	042	Clear Block of Storage
	043	Clear 32K
storage fault type 1100/80A	035	Down CPU
	036	Down SIU Segment
	037	Down MSU Bank
	040	Down 64-Word Block
	041	Down Then Up 64-Word Block
	042	Down CPU-SIU I/Face
	043	Down Cluster
	044	Down Chip
	045	Down SIU Half
	046	Clear Block of Storage
	047	Clear 32K
storage fault type 1100/60	050	Down CPU
	051	Down SIU Segment
	052	Down MSU Bank
	053	Down 64-Word Block
	054	Down 64-Word Block, Then Up Block
	055	Down CPU - SIUSEG Interface
	056	Down Cluster
	057	Down Chip
	060	Down SIU Half
	061	Select Block of Storage and Clear It
	062	Select 32K of Storage and Clear It
	063	Down MSP of 1100/60 Storage
	064	Up CPU
	065	Down CPU Then Up CPU
	066	Down CPU (Fatal) - SSP Decides
	067	CS Correction Completion - FK Complex - I/F
	070	CS Not Correction - FK Complex - I/F
	071	SSP Did Not Receive IMI Request For CS Correction - Interface Busy

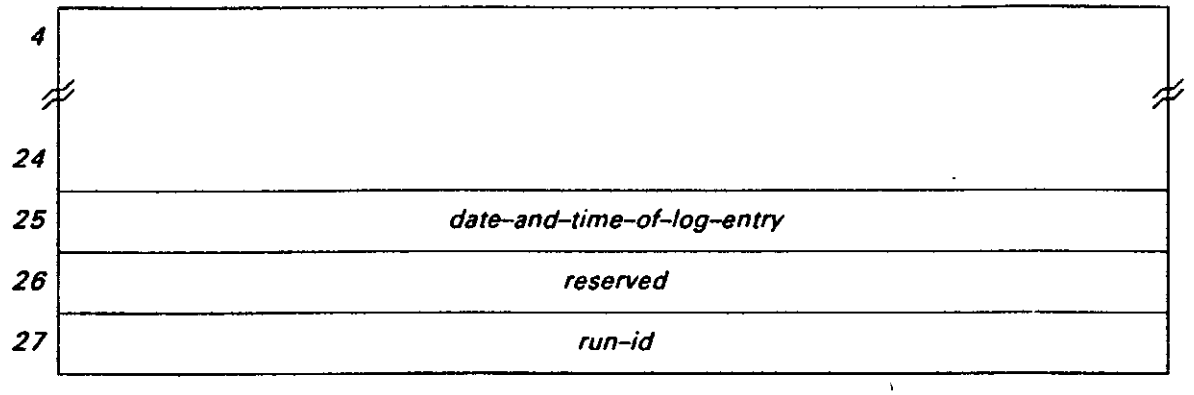
9.28. Common Bank Reload Log Entry

Whenever a common bank is reloaded, a log entry is created. The request type describes why the common bank was reloaded. The format of this log entry is shown in Table 9-43 and described below.

Table 9-43. Common Bank Reload Log Entry

0	*entry type (28)*	*nbr-of-wds in-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wd-blk*
1	*bank-name*					
2						
3				*request-type*		

Table 9–43. Common Bank Reload Log Entry (continued)



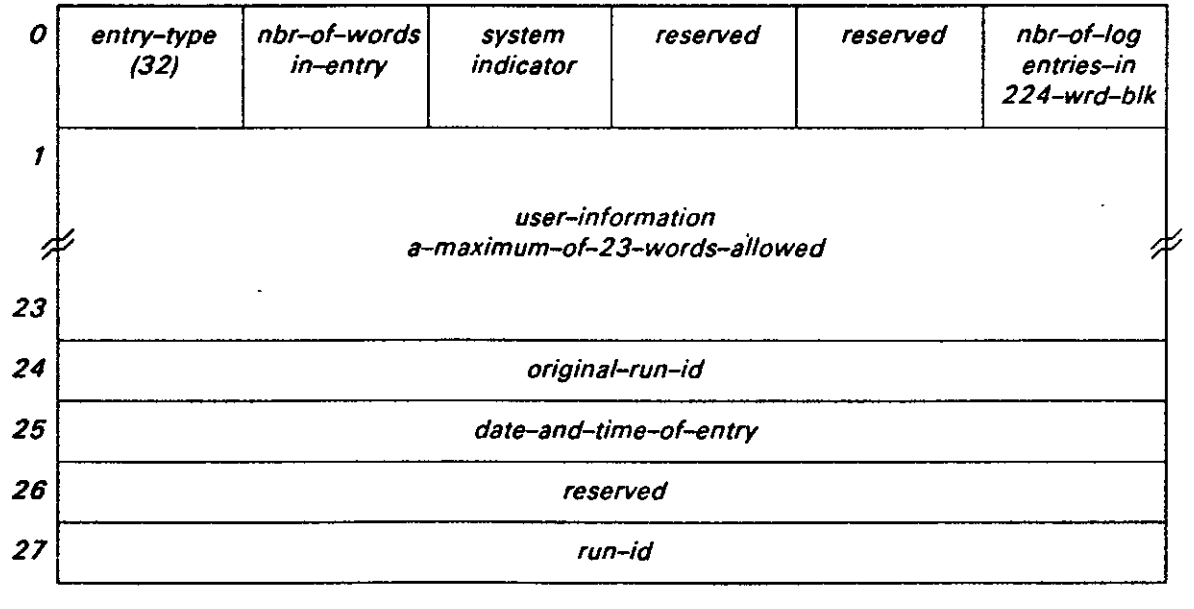
Word 3

request-type	0001	immediate request (ER)
	0002	error request (ER)
	0004	stall request (ER) register input RL BDI based
	0010	stall request (ER) buffer input RL BDI based
	0020	stall request (ER) register input RL BDI not based
	0040	stall request (ER) buffer input RL BDI not based
	0100	immediate request (console)
	0200	error request (console)
	0400	stall request (console)

9.29. Log Entry Created by ER LOG$

This log entry is created by the user via ER LOG$ for insertion in the SYS$*SYSTEM$LOG$ file. The format is shown in Table 9–44.

Table 9–44. User Formatted Log Entry



NOTE: In this log entry, nbr-of-words-in-entry (S2 Word 0) contains only the number of words in the user-information. (Words 1 through 23)

9.30. Checkpoint/Restart Termination

The checkpoint/restart termination entry is shown in Table 9-45.

Table 9-45. Checkpoint/Restart Termination Entry

0	*entry-type (33)*	*nbr-of-wrds in-the log-entry*	*system indicator*	*reserved*	*reserved*	*nbr-of-log entries-in 224-wrd-blk*
1	*primary-storage-reference-count*					
2	*primary-storage-reference-count*					
3	*extended-storage-reference-count*					
4	*extended-storage-reference-count*					
5	*ER-and-control-card-charge*					
6	*I/O-SUPS*					
≈						
15						
16	*voluntary-delay-time*					
17	*CPU-time*					
18						
≈						
24						
25	*date-and-time-of-last-entry*					
26	*reserved*					
27	*run-id*					

9.31. Security Log Entry

Security log entry type 34 for security reasons is located in the Security Officer Guide Manual.

9.32. Log Buffer Entries for IIIX

The log buffer entries for IIIX failures is shown in Table 9-46.

Table 9-46. Type 37 Log Buffer Format for IIIX Failures

0	type (37)	nbr of words in entry	system indicator	reserved	nbr of log entries in 224-wrd blk
1				subcode	
2	sending CPU number	receiving CPU number	unused	return address of IIIX requestor	
3	number of retry attempts (negative if all failed)				
4	action requested of IIIX interrupt handler (contents of IIIQUE)				
5	return address of IIIFRT caller				
6					
≈					
24					
25	time and date				
26	reserved				
27	EXEC 8				

Word 1

subcode Subcode for the log entry; values are:

2 - logged by FLTLOG following an XER 015

1 - logged due to CPU initialization failure in element INIT

0 - logged while EXEC is running, following system initialization

Word 2

Contents of IIISAV cell.

9.33. Hardware Monitor Log Entry (1100/60)

One such log entry is created for each CPU up every time hardware monitor data is retrieved (approximately every 25 seconds) on 1100/60 Systems that have a hardware monitor. The format differs depending on the expansion type of the IOU. The format for each expansion type is shown in Tables 9-47 through 9-49.

Table 9-47. Hardware Monitor Log Entry (No Expansion)

0	entry-type (39)	nbr-of-words in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					status	subtype
2					IOU 1	IOU Ø
3		nbr-of CPUs[2]	CPU nbr	SSP-time-stamp		
4		sample-counter		CPU-idle (0111)		
5		guard-mode		I/O word channel A I/F Ø active		
6		I/O word channel A I/F 1 active		I/O word channel A I/F 2 active		
7		I/O word channel A I/F 3 active		not used		
8		not used		not used		
9		not used		not used		
10		not used		not used		
11		not used		I/O block Mux A active		
12		not used		EXEC software state 1 (0000)[1]		
13		EXEC software state 2 (0001)[1]		EXEC software state 3 (0010)[1]		

Table 9-47. Hardware Monitor Log Entry (No Expansion) (continued)

14		EXEC software state 4 (0011)[1]			EXEC software state 5 (0100)[1]	
15		EXEC software state 6 (0101)[1]			EXEC software state 7 (0110)[1]	
16		real time (1000)			ESI completion (1001)	
17		user software state 3 (1010)[1]			user software state 4 (1011)[1]	
18		TIP (1100)			Deadline batch (1101)	
19		Demand (1110)			Batch (1111)	
20	physical to logical channel module correspondence information IOU Ø see format of next words					
21	IOU 1 Module 5	Channel Module 4	Channel Module 3	Channel Module 2	Channel Module 1	Channel Module Ø
22	unused					
23	unused					
24	unused					
25	date-and-time-of-log-entry (TDATE$)					
26	reserved					
27	EXEC 8					

Table 9-48. Hardware Monitor Log Entry (Expansion Type F-2688)

0	entry-type (39)	nbr-of-words in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					status	subtype
2					IOU 1	IOU Ø
3		nbr of CPUs [2]	CPU nbr	SSP-time-stamp		
4		sample-counter		CPU-idle (0111)		
5		guard-mode		I/O word channel A I/F Ø active		
6		I/O word channel A I/F 1 active		I/O word channel A I/F 2 active		
7		I/O word channel A I/F 3 active		I/O word channel B I/F Ø active		
8		I/O word channel B I/F 1 active		I/O word channel B I/F 2 active		
9		I/O word channel B I/F 3 active		I/O word channel C I/F Ø active		
10		I/O word channel C I/F 1 active		I/O word channel C I/F 2 active		
11		I/O word channel C I/F 3 active		I/O block Mux A active		
12		I/O block Mux B active		EXEC software state 1 (0000)[1]		
13		EXEC software state 2 (0001)[1]		EXEC software state 3 (0010)[1]		
14		EXEC software state 4 (0011)[1]		EXEC software state 5 (0100)[1]		
15		EXEC software state 6 (0101)[1]		EXEC software state 7 (0110)[1]		
16		real time (1000)		ESI completion (1001)		
17		user software state 3 (1010)[1]		user software state 4 (1011)[1]		
18		TIP (1100)		Deadline batch (1101)		
19		Demand (1110)		Batch (1111)		

Table 9-48. Hardware Monitor Log Entry (Expansion Type F-2688) (continued)

20	physical to logical channel module correspondence information IOU Ø see format of next words					
21	IOU 1 Module 5	Channel Module 4	Channel Module 3	Channel Module 2	Channel Module 1	Channel Module Ø
22	unused					
23	unused					
24	unused					
25	date-and-time-of-log-entry (TDATE$)					
26	reserved					
27	EXEC 8					

Table 9-49. Hardware Monitor Log Entry (Expansion Type F-2916)

0	entry-type (39)	nbr-of-words in-entry	system indicator	reserved	reserved	nbr-of-log entries-in 224-wrd-blk
1					status	subtype
2					IOU 1	IOU Ø
3		nbr of CPUs[2]	CPU nbr	SSP-time-stamp		
4		sample-counter		CPU-idle (0111)		
5		guard-mode		I/O word channel A I/F Ø active		
6		I/O word channel A I/F 1 active		I/O word channel A I/F 2 active		
7		I/O word channel A I/F 3 active		I/O word channel B I/F Ø active		
8		I/O word channel B I/F 1 active		I/O word channel B I/F 2 active		
9		I/O word channel B I/F 3 active		I/O block Mux C active		
10		not used		not used		
11		not used		I/O block Mux A active		
12		I/O block Mux B active		EXEC software state 1 (0000)[1]		

Table 9-49. Hardware Monitor Log Entry (Expansion Type F-2916) (continued)

13		EXEC software state 2 (0001)[1]			EXEC software state 3 (0010)[1]	
14		EXEC software state 4 (0011)[1]			EXEC software state 5 (0100)[1]	
15		EXEC software state 6 (0101)[1]			EXEC software state 7 (0110)[1]	
16		real time (1000)			ESI completion (1001)	
17		user software state 3 (1010)[1]			user software state 4 (1011)[1]	
18		TIP (1100)			Deadline batch (1101)	
19		Demand (1110)			Batch (1111)	
20	physical to logical channel module correspondence information IOU Ø see format of next words					
21	IOU 1 Module 5	Channel Module 4	Channel Module 3	Channel Module 2	Channel Module 1	Channel Module Ø
22	unused					
23	unused					
24	unused					
25	date-and-time-of-log-entry (TDATE$)					
26	reserved					
27	EXEC 8					

status	0	data normal
	1	data missing
	2	data overflow
IOU	0	no expansion
	1	expansion type F-2688
	2	expansion type F-2916

The subtype can take on values of 0 or 1 and is used only as a sequential index for a given hardware monitor retrieval. The time stamp is a three byte (8 bits/byte) stamp in hours, minutes, and seconds past midnight. The 4 digit binary notation indicates which d-bits are set to define the given software states. D-bits 2, 24, 25, and 26 correspond to the most significant through least significant digits shown.

1 These states not currently available.

2 Defined only in subtype = 0 entries.

Glossary

A

absolute element

An element containing a complete program in binary form suitable for execution by the Executive. Such elements normally occur as output from a collection of relocatable elements, with all necessary linkages and relocation performed.

ASCII

An 8-bit character code that is used for Executive files and user interfaces. Refer to USA Standard X3.4-1967.

auxiliary storage

Supplemental storage, not directly addressable by a CPU and accessible only via I/O (as opposed to main storage) and typically having much greater capacity. Includes magnetic tape, flying-head magnetic drum, sector formatted drum, or disk.

B

bit

Binary digit. The fundamental unit of storage having the value 0 or 1. Bits are grouped in bytes and words to form the functional manipulative units of storage devices.

breakpoint (symbiont)

Division of symbiont-defined files into parts such that the output of completed parts may be initiated prior to run completion. This procedure allows more efficient utilization of printers and punches when large symbiont output files are involved.

breakpoint (hardware register)

See "programmed breakpoint".

byte

A group of adjacent bits usually operated upon as a unit; can be 6, 9, 12, or 18 bits.

C

cataloged file

A file known to and retained by the Executive, for an indefinite period not necessarily related to the life of a particular run, and generally retrievable by runs other than the run which originally created the file. In some cases, a cataloged file may be accessed simultaneously by two or more runs.

collection

The process by which individual (relocatable) elements are combined to form a complete program (absolute element). This process begins with explicit specification of elements to be included, and typically involves inclusion of additional unspecified elements required to satisfy undefined references (these are most commonly obtained from the system relocatable subroutine library file SYS$*RLIB$).

Collector

A system processor that provides the collection function.

control statement

A data image used to direct the Executive in processing a run. A control statement is identified by a master space or at symbol (@) in column 1.

cycle

A number used to differentiate successive updates of files or symbolic elements.

D

data image or image

A data record in human intelligible (character) format; most commonly refers to punched card or printer data, or communications terminal input/output records.

E

element

A named grouping of information typically manipulated as a unit, and typically defining a logical program part such as a subroutine. There are four basic types of elements: symbolic, relocatable, absolute, and omnibus. Elements are parts of program files or element files.

element file

A specially structured file containing a group of elements, residing on magnetic tape, as opposed to a program file. The arbitrary distinction between the two file types is made to avoid confusion between operations that may be done on one medium but not the other. Element files are created by the @COPOUT command and read by @COPIN.

EXEC or Executive

Series 1100 Executive System. An Executive is a routine that controls the execution of other routines. The Executive is the principal interface between the user and the system as a whole. It is responsible for such functions as time and space allocation of system resources; first-level I/O control and interrupt answering; logging of system accounting data; first-level debugging assistance; and protection against undesired interaction of users with other users of the system.

F

Fieldata

A 6-bit character code that is the internal character set of the EXEC.

file

An organized collection of data, treated as a unit, and stored in such a manner as to facilitate the retrieval of each individual datum.

fixed mass storage

Drum, unitized channel storage, sector-formatted drum units, and disk units declared to be fixed during the boot of the system. This storage is considered to be permanent (online).

G

granule

The incremental unit of size in which a storage medium can be allocated.

I

I/O

Input/Output. The process of transferring information between the central processor and peripheral devices. I/O devices include: magnetic tapes, magnetic disks, magnetic drums, CTSs, card readers, printers, and punches.

L

LJSF

Left-justified and space-filled.

language processor

A processor whose principal functions include compiling, assembling, translating, interpreting, or related operations for a specific programming language (e.g., COBOL, FORTRAN, Assembler, etc.), as opposed to a system processor.

M

main storage

Main storage provides storage for instruction and data words accessible to the processors. In systems with an SIU, the main storage is accessed through the SIU.

mass storage

Auxiliary storage with random access capability; includes all types of auxiliary storage with the exception of magnetic tape.

Master File Directory (MFD)

A directory maintained by the Executive to control the retrieval and retention of cataloged files.

mnemonic

Word or term devised to aid the human memory. Includes acronyms, such as TTY (teletypewriter), and error mnemonics, such as PWRLOS (power loss).

O

omnibus element

An element of arbitrary format used to store general information in a program file.

operating system

The Series 1100 Operating System. The entire set of system software available for the Series 1100 which is either a part of, or operates under, the Executive System. This includes the Executive System proper, compilers, utility programs, subroutine libraries, and so forth.

P

PCT

Program Control Table. A special table maintained by the Executive containing the bulk of the control information for a particular run and the program (if any) currently in execution for that run.

processor call statement

A control statement used to direct the execution of a processor. Such statements have a standardized format that facilitates specification of parameters typically required by processors, such as element names.

program

Generally a series of instructions, in a form acceptable to a computer, prepared in order to achieve a certain result. In the context of run processing, a program is an absolute element to be executed as a task; may be a processor or a user program.

program file

A partitioned random access file consisting of a group of elements residing on mass storage. A program file may contain symbolic, relocatable, omnibus, or absolute elements or a combination of elements.

programmed breakpoint

On 1100/60, and 1100/80 Systems, conditioning of a special CPU register such that an interrupt occurs upon reading, writing, or execution of a specific location or group of locations. (See *1100/60 Systems, System Support Processor (SSP), Operator Reference*, UP-9123 (applicable version.)

R

RJSF

Right justified and space filled.

relocatable binary format

This is the format of the output of the language processors, e.g., FTN, ACOB, and MASM. Relocatable binary format may also be produced by the Series 1100 Collector.

relocatable element

An element containing a program part in relocatable binary format, suitable for combination with other relocatable elements to produce an executable program (absolute element). Such elements occur most commonly as the output of a language processor to be input to a collection.

run

A specified group of tasks prescribed as a unit of work for the system. The run is the largest work grouping treated and manipulated as a unit by the Executive. The tasks of a particular run are executed serially in the order specified by the runstream.

runstream

A sequence of data images that, taken as a whole, constitute the specification of a run. A runstream consists of an @RUN control statement, followed by other control statements and data, which direct the performance of individual tasks.

S

SDF

System Data Format. The standard data format employed by the operating system. Briefly, SDF is a sequential, fixed-block, variable-record format in which records may span blocks.

sector-formatted mass storage

Mass storage accessible in units of 28 words (one sector). This is the most common mass storage format used on Series 1100 Systems.

software

A set of computer programs including the operating system and user programs, as opposed to hardware.

swapping

The process of storing low-priority or suspended programs on mass storage to allow main storage space to load other higher priority programs.

switching

The process by which the Executive controls usage. This principally involves determination of which activities of which programs are to be executed on which CPU(s) for how long, and the control functions needed to fulfill that determination. This is also called dispatching.

symbolic element

An element containing information generally in human-intelligible format (typically card images). The most common use of symbolic elements is as source language to be input to a language or system processor.

symbionts

A complex of Executive routines providing the user interface with unit record peripherals; e.g., onsite and remote card readers and printers.

symbiont devices

Relatively slow-speed devices, such as card readers, card punches, and printers which are controlled by symbionts and are used to provide direct input to and output from the system.

system

The total Series 1100 hardware/software complex comprising an integrated information processing installation.

T

temporary file

A transient file created by, accessible to, and existing within the life of, a single run only.

Temporary Program File (TPF$)

A mass storage file assigned automatically by the Executive to each run. As a convenience to the user, in a wide variety of program file and element manipulation operations TPF$ is assumed as the program file in the absence of an explicit filename reference.

track

In the context of sector-formatted mass storage, a granule consisting of 64 sectors, each sector consisting of 28 words, giving a total of 1792 words per track. For word-addressable mass storage, a granule consisting of 1792 words.

U

user

An individual or organization that consumes services provided by the system.

user–id

Identifies a particular user (person) of the system. A 12–character string that represents a person's identity.

user program

Any program other than a processor. Usually developed by a user; however, certain system software packages operate as free–standing user programs (e.g., FMPS).

W

word

A sequence of bits or characters treated as a unit and capable of being stored in a single main storage location. (A word is represented by 36 bits for Series 1100 Systems.)

word–addressable

Mass storage which is capable of being accessed in units of single words. This is most efficient on hardware having this capability (i.e., flying head magnetic drum), but for other cases it is simulated by the Executive.