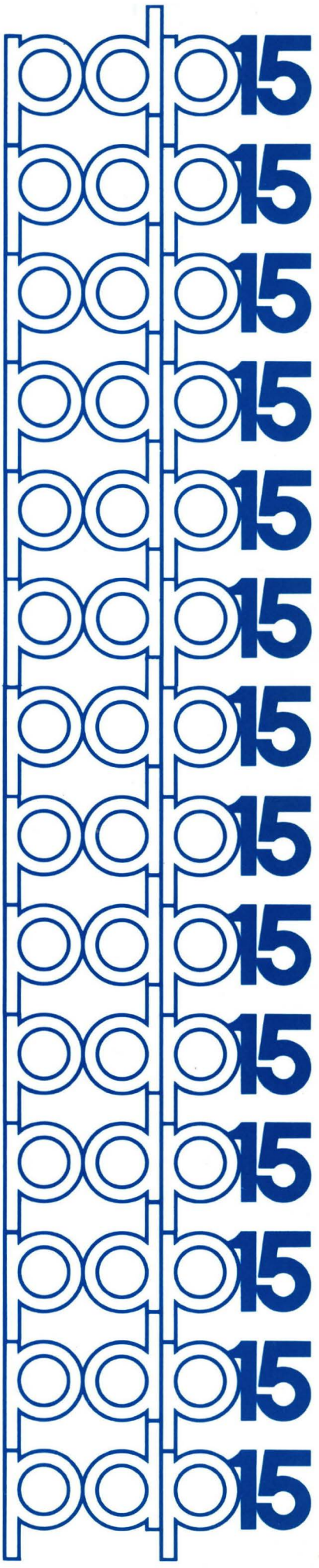
digital

linking loader

utility program

digital equipment corporation



DEC-15-YWZB-DN8

LINKING LOADER

UTILITY PROGRAM

For additional copies, order DEC-15-YWZB-DN8 from the
Program Library, Digital Equipment Corporation, Maynard,
Massachusetts, 01754 Price $2.00

Revised February, 1972

The material in this manual is
for information purposes and is
subject to change without notice.

PREFACE

This manual describes the operation and use of the Linking Loader Utility Program. The Linking Loader program may be operated in either the ADVANCED Software System (ADSS) or the Disk Operating System (DOS) environment.

It was assumed in the preparation of this manual that the reader was familiar with the operation of the PDP-15 equipment and the contents of the software manual describing the features of the particular monitor system in which he was operating, that is:

a) for ADSS users, PDP-15/20/30/40 ADVANCED Monitor Software System Manual, DEC-15-MR2B-D;

b) for DOS users, DOS Software System User's Manual, DEC-15-MRDA-D.

PDP-15 UTILITY PROGRAMS MANUAL, DEC-15-YWZB-D

The PDP-15 Utility Programs manual is comprised of a set of individual manuals, each of which describes the operation and use of a PDP-15 Utility Program. The manuals which make up the Utility Programs set are listed in the following Application Guide. In addition, the Application Guide also indicates the order number of each manual and the specific PDP-15 Monitor Software Systems in which the program described may be used.

The Utility Manuals may be ordered either individually, by using the title and order number given with each manual, or as a set, by referencing "PDP-15 Utility Programs Manual, DEC-15-YWZB-D".

APPLICATION GUIDE

PDP-15 UTILITY PROGRAM MANUALS

PDP-15 Utility Program Manuals and the Application of Each

Title	Manual Order Number (DEC-15-YWZB_)	Applies to Monitor:		
		DOS	ADV	B/F
DDT Utility Program	DN1	✓	✓	✓
CHAIN & EXECUTE Utility Program	DN2	✓	✓	✓
SGEN ADVANCED Monitor	DN3		✓	
MTDUMP Utility Program	DN4	✓	✓	
PATCH Utility Program	DN5	✓	✓	✓
EDIT Utility Program	DN6	✓	✓	✓
UPDATE Utility Program	DN7	✓	✓	✓
LINKING LOADER	DN8	✓	✓	✓
PIP ADVANCED Monitor	DN9		✓	✓
SRCCOM Utility Program	DN11	✓	✓	✓
SGEN DOS Monitor	DN12	✓		
PIP DOS Monitor	DN13	✓		

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 GENERAL DESCRIPTION

The Linking Loader is a program which operates under the DOS-15, ADVANCED and Background/Foreground Monitor Systems of the PDP-15. The Linking Loader loads and links both relocatable and absolute[1] binary program units as output from either the FORTRAN IV Compiler or MACRO-15 Assembler. These program units consist of machine language instruction codes and special "loader codes" which tell the loader how to load the program. These program units can reside on the input device not only as separate files, but also as library files. The structure of the data input and description of the loader codes are provided in Chapter 2. Figure 1-1 illustrates the I/O functions of the Loader.

Initially, the Loader loads all the program units named in the command string (see Operating Procedures, Chapter 2). The Loader then automatically loads and links all requested I/O handlers and library subprograms which have been globally linked[2]. The requested library subprograms are loaded from the external (user) library (if one exists) and the system library (in that order). After both libraries have been examined for requested subprograms, the Loader prints the names of all subprograms which have not been found. I/O handlers that are already in core for the Loader's use will be retained if required for the user's program. The Loader also assigns COMMON data storage areas for FORTRAN IV program units. Individual program units cannot be executed if the program flows across a 4K page, or 8K bank. The Loader prevents this type of loading, but will load (and link) the unit into the next memory bank. No overlap checking of any kind is made with absolute binary program units.

Optionally, symbols and their absolute definitions are loaded into a program dictionary (symbol table) for use by the DDT (dynamic debugging

---

[1] A MACRO assembled program headed by a .LOC statement, e.g., .LOC 100, is an absolute binary program and the binary is output in link loadable format. A program headed by an .ABS, .ABSP, .FULL, or .FULLP statement is output as absolute block binary and cannot be loaded by the Linking Loader.

[2] Global or symbolic linkages are established through the use of .GLOBL pseudo-ops in MACRO-15 and the external function references and CALL statements in FORTRAN.

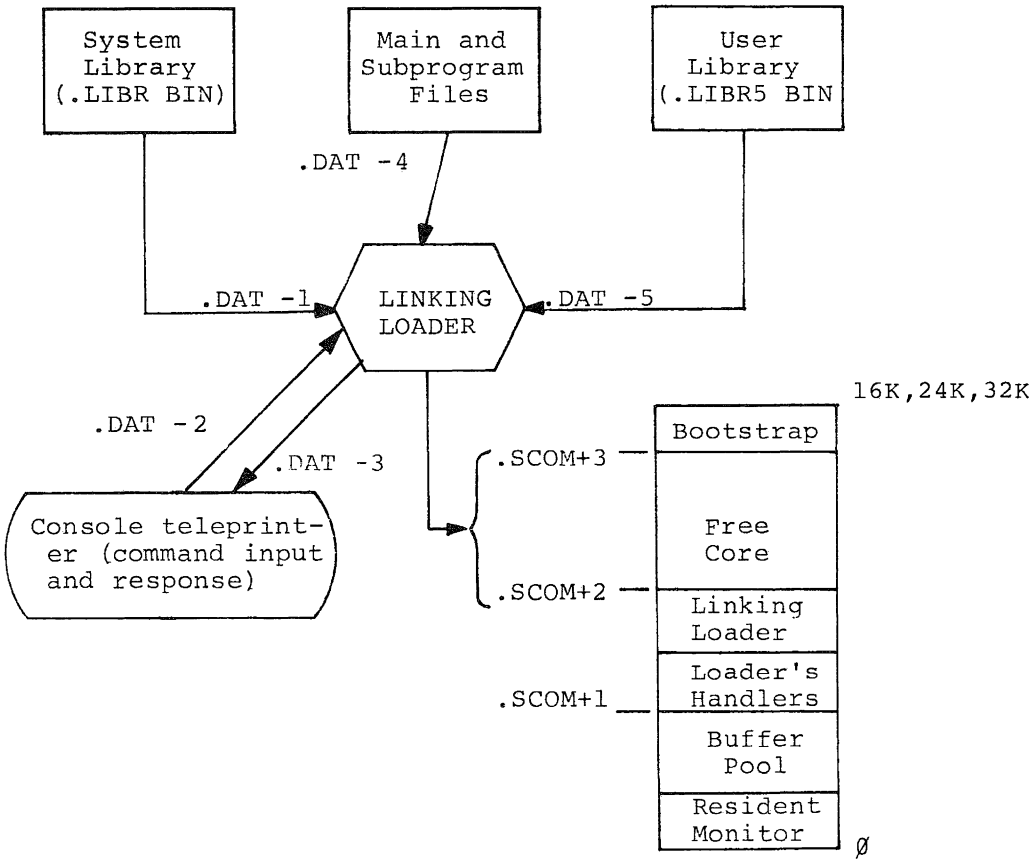


Figure 1-1

Linking Loader I/O Function

technique) Utility Program. The Loader also sets up, for use by DDT, the start execution address of the main program (in the System Communication Table) and the initial relocation value of all the program units (in the symbol table). Refer to DEC-15-YWZB-DN1 for further information about DDT.

1.2 FORTRAN COMMON STATEMENTS

The Linking Loader permits FORTRAN COMMON blocks and block-data subprograms to overlap memory pages and banks. When operating in either Bank or Page Mode, the Loader allows COMMON block sizes greater than

8192_{10}, provided that each element in COMMON does not exceed 8192_{10}. For example, the statement

```
COMMON  /I/L(100,100)
```

is illegal because the size of array L is 10000_{10}.

However, the statement:

```
COMMON  /I/L1(100,50),L2(100,50)
```

is acceptable. Each array size is 5000_{10} and the size of the COMMON block is 10000_{10}.

Non-COMMON arrays and variables are initialized to zero by the Loader.

MACRO programs can be linked to COMMON areas defined by FORTRAN IV. If any unresolved globals remain after the Loader has searched the user and system libraries and has defined COMMON blocks, the Loader tries to match those global names to COMMON block names. If a match is made, the global becomes defined as the COMMON block. For example:

```
FORTRAN IV PROGRAM
  INTEGER A,B,C
  COMMON/NAME/C
  COMMON A,B
    .
    .
    .
MACRO PROGRAM
  .GLOBL NAME,.XX          /.XX IS NAME GIVEN TO BLANK COMMON
                           /BY THE F4 COMPILER
  DZM* .XX                 /CLEAR A - NOTE INDIRECT REFERENCE
  ISZ  .XX                 /BUMP COUNTER
  DZM* .XX                 /CLEAR B
  DZM* NAME                /CLEAR C
```

Note that if the values are REAL (2 words) or DOUBLE PRECISION (3 words) the MACRO program must account for the number of words when accessing specific variables.

1.3 RELATED REFERENCE MATERIAL

The manuals listed below contain information which is necessary in understanding and using the Linking Loader.

a. DOS-15 System Manual - DEC-15-NRDA-D
   DOS-15 User's Manual - DEC-15-MRDA-D
   DOS-15 Keyboard Command Guide - DEC-15-NGKA-D

b. ADVANCED Monitor Systems
   ADVANCED Monitor Software System for
      PDP-15/20/30/40 - DEC-15-MR2B-D
   PDP-15/20 User's Guide - DEC-15-MG2B-D
   UPDATE Utility Program Manual - DEC-15-YW2B-DN7
   MACRO-15 Assembler Manual - DEC-15-AM2C-D
   FORTRAN IV Language Manual - DEC-15-GFWA-D

1.4 SPECIAL SYMBOLS

The following symbols, when used, are defined as follows:

Symbol	Meaning
↵	Carriage RETURN
→|	CTRL TAB
␣	Space
[ ]	Optional Command Element
{ }	One of the enclosed command elements must be chosen.

CHAPTER 2

LOADER CODE DESCRIPTIONS

As mentioned in Chapter 1, the relocatable and absolute binary program units output by the FORTRAN IV Compiler and the MACRO-15 Assembler contain both machine language instructions and loader codes. These codes are assigned by FORTRAN and MACRO to identify the various elements of the binary program. The loader, in turn, interprets these codes to properly relocate, link, assign COMMON areas, preserve constants, etc. The paragraphs which follow provide descriptions of the physical organization of relocatable program units and library files and definitions of the loader codes.

2.1 INFORMATION UNITS

The binary output from the FORTRAN compiler and the MACRO Assembler consists of named files containing blocks of information units. Each information unit consists of loader code (6 bits) and a data word (18 bits). The form of the object program at run time is determined by the content and the ordering of the information units. Several information units may be grouped to convey a single run-time instruction to the Loader.

Information units are grouped in blocks of four 18-bit machine words as shown in Figure 3-1.

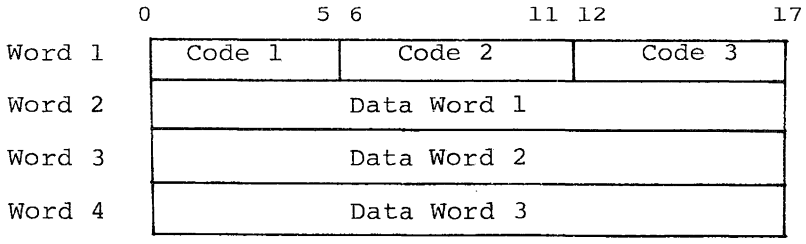


Figure 2-1

Information Unit Block Structure

IOPS binary records of 48 information words and a 2-word header are accepted by the loader.

2. 2 PROGRAM UNIT ORGANIZATION

A program unit consists of as many information units as are required to contain the binary program. The two basic types of program units are diagrammed in Figures 2-2 and 2-3.

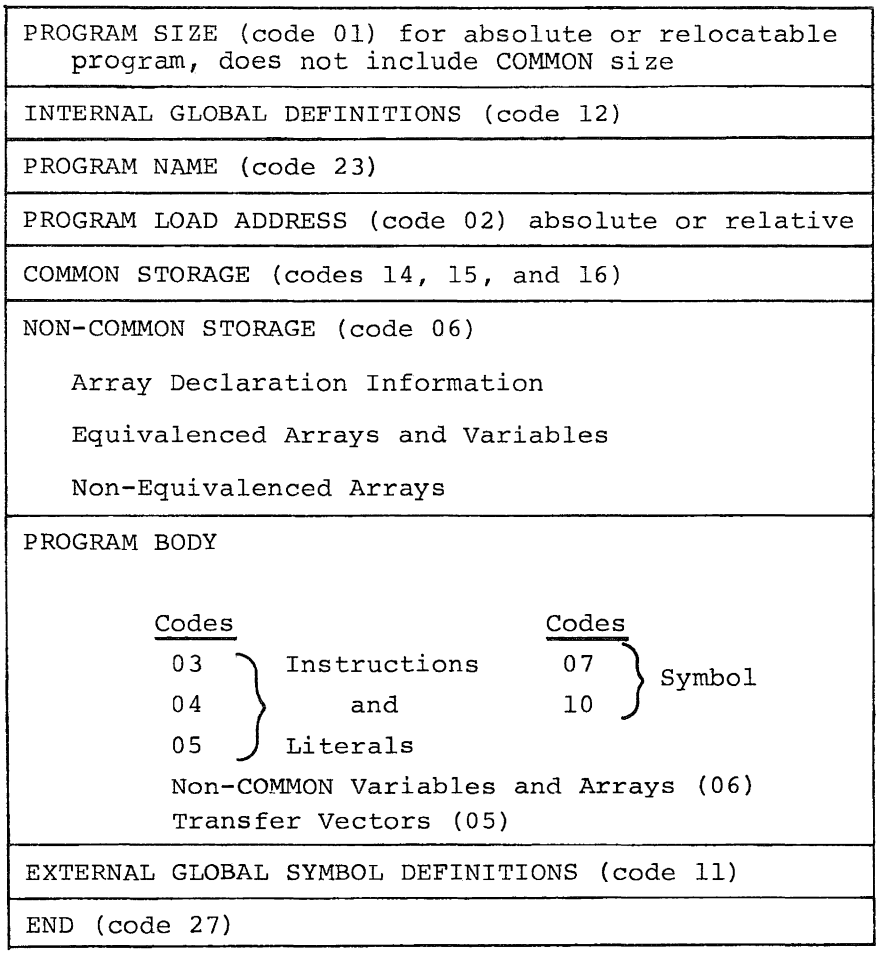


Figure 2-2

Main Program and Subprogram Organization

BLOCK DATA INDICATOR (code 13)
PROGRAM NAME (code 23)
COMMON STORAGE (codes 14, 15, and 16)
DATA INITIALIZATION CONSTANTS (codes 17, 20, 21, and 22)
END (code 27)

Figure 2-3

Block Data Subprogram Organization

2.3 LIBRARY FILE ORGANIZATION

Both system and user library files are structurally identical and are created and maintained by the UPDATE Utility Program (described in DEC-15-YWZA-DN7). A library file, unlike other files, consists of a number of program units (rather than just one) in which all end-of-file codes, except the last, have been removed. Figure 2-4 shows the complete structure of a library file.

2.4 IDENTIFICATION CODES

The identification code contained in each information unit tells the Loader how to interpret the associated data word. As mentioned earlier, there is an implied order in which codes appear within a binary file.

Code	Loader Action
01	Program Unit Size The data word specifies the number of machine words required by this program unit. This number does not include the required number of machine words for COMMON storage. The program size is used by the Loader to determine whether or not the program will fit within the unused locations of any available page or bank. This information unit appears only once per program unit and is the first information unit of the binary output. In absolute loads, no checking is made to prevent overlay of other program units; this is left to the user. The program size is also used to determine where

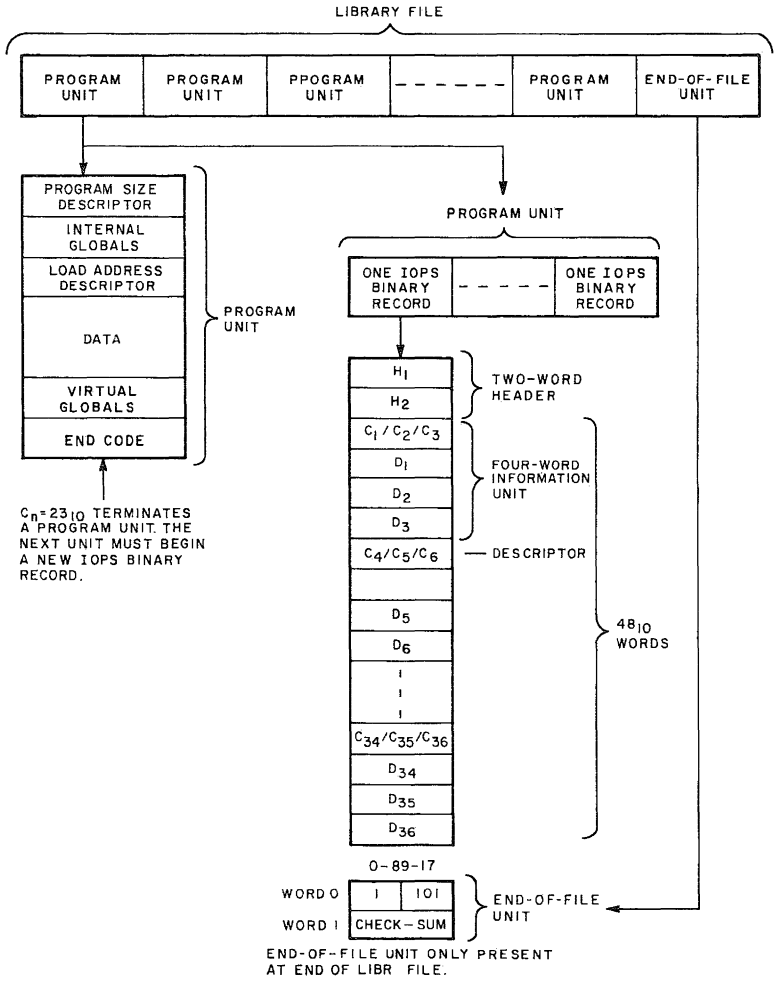


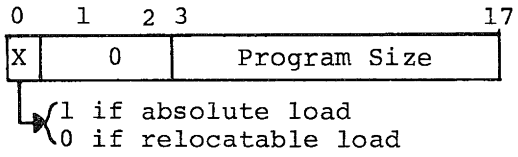
Figure 2-4

Library File Organization

Code | Loader Action

to begin loading as loading proceeds from the top of core down.

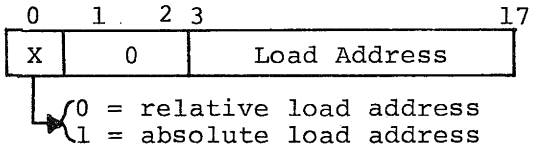
Data Word



02 Program Load Address

The data word is an unrelocated memory address. This address specifies either an absolute or a relative storage address for program data words and is incremented by one for each data word stored (codes 03, 04, and 05). If the address is relative, it is initially incremented by the current relocation factor (modulo 15 bits). Bit 0 of the data word is used to indicate an absolute address (bit 0 = 1) or a relative address (bit 0 = 0).

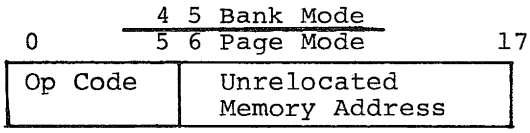
Data Word



03 Relocatable Instruction

The data word is a memory referencing instruction. The address portion of the instruction is incremented by the current relocation factor (modulo 12 bits for page mode and 13 bits for bank mode). The instruction is stored in the location specified by the load address which is incremented by one after the word is stored.

Data Word

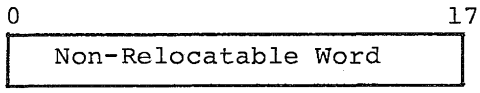


04 Absolute Instruction/Constant/Address

The data word is either a non-memory referencing instruction, a non-relocatable memory referencing instruction, an absolute address, or a constant. The word is stored in the location specified by the load address which is incremented by one after the word is stored.

Code	Loader Action

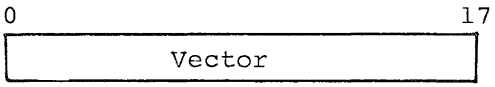
Data Word



05 Relocatable Vector

The data word contains a relocatable program address (vector). The word is incremented by the current relocation factor (modulo 15 bits). The data word is stored in the the location specified by the load address which is incremented by one after the word is stored.

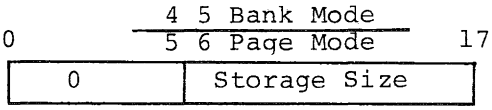
Data Word



06 Non-COMMON Storage Allocation

The data word specifies the number of machine words required for non-COMMON variable and array storage. Storage allocation begins at the address specified by the load address. The load address is incremented by this number. This block of memory is cleared.

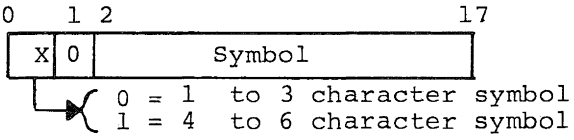
Data Word



07 Symbol - First Three Characters

The data word contains the first three characters of a symbol in radix 50_8 format (see Appendix C). The data word is saved by the Loader for future reference.

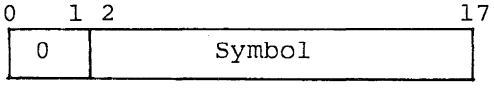
Data Word



10 Symbol - Last Three Characters

The data word contains the last three characters of a symbol in radix 50_8 format. The data word is saved by the Loader for future reference. This word is used only if in the code 07 data word bit 0 = 1.

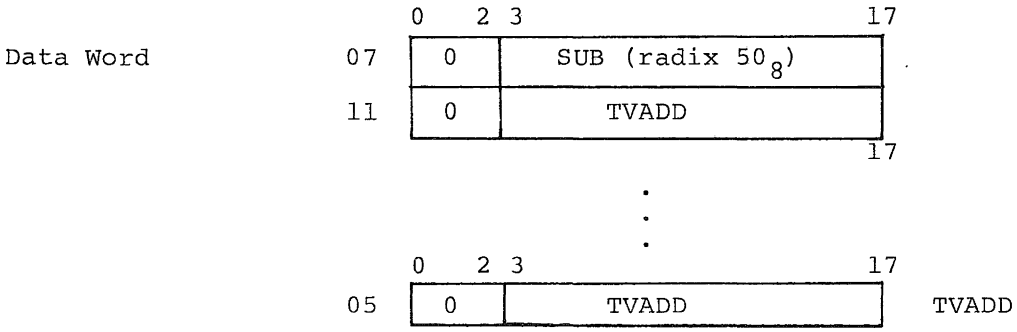
Data Word



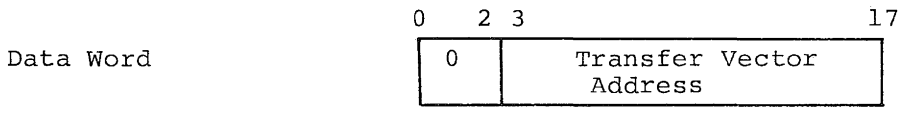
Code	Loader Action

11 External Symbol Definition

The data word contains the unrelocated address of the transfer vector for the subprogram named by the last symbol loaded (codes 07 and 10). If the external subprogram has already been loaded, the address (definition) of the symbol is stored into the specified vector address (relocated modulo 15 bits). If the subprogram has not been loaded and this is the initial request, the symbol and the relocated (modulo 15 bits) transfer vector address are entered into the Loader symbol dictionary as a request for subprogram loading. This action automatically forces the Loader into a Library Search Mode when the end of the command string is encountered. If the Loader is already in the Library Search Mode, it remains there until all unresolved globals have been resolved. If the subprogram has been previously requested (symbol in dictionary) but not loaded, the Loader chains the reference locations. This chain, generated exclusively by the Loader, is followed when the external definition is encountered. (Unchained transfer vector locations must initially contain a reference address (code 04 or 05) to themselves.) For example, .GLOBL SUB where SUB is virtual causes the output of the following:



SUB is defined internally as TVADD. Subroutine calls are made via JMS* SUB.

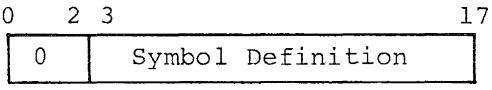


Code	Loader Action

12 Internal Global Symbol Definition

The data word contains the unrelocated or absolute address (definition) of the last symbol loaded (codes 07 and 10). The last symbol loaded is a global symbol internal to the program unit which follows. In the Library Search Mode, if a request for subprogram loading exists (code 11) in the Loader dictionary, the relocatable (modulo 15 bits) or absolute definition is stored in the specified transfer vectors and the program unit is loaded. The definition also replaces the transfer vector address in the Loader dictionary. If no request for loading exists, the program unit is not loaded and the Loader continues to examine information units until the next internal global symbol definition is found (Library Search Mode). If the program unit is to be loaded, all internal global symbols following the one causing loading are automatically entered into the Loader dictionary as defined global symbols. If the symbol already exists in the dictionary and is defined (indicating that a program unit with the same name is already loaded) the Loader does not try to load the program unit again.

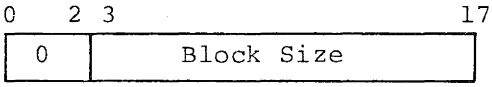
Data Word



13 Block Data Declaration

This information unit instructs the Loader that the COMMON blocks and data constants following are part of a block data subprogram.

Data Word



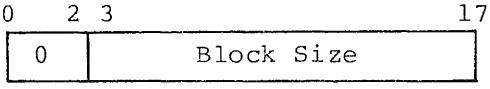
14 COMMON Block Definition

The data word specifies the number of storage words required for the COMMON block named by the last symbol loaded (codes 07 and 10). In general, the assignment of memory space for the COMMON block is deferred until all requested library subprograms have been loaded. The exception to this rule occurs when the block data declaration (code 13) has been encountered. In this case, the COMMON block name is treated as an internal global symbol, and the block is

assigned to memory. After the block is assigned to memory, the starting address is entered into the Loader dictionary, and the starting address is saved by the Loader for future use (code 15). All symbols in the dictionary associated with the block are assigned addresses with respect to this starting address. All symbols which are yet to be loaded (via code 15 and 16) will also be assigned as they are encountered. When the block data flag is not set, the Loader enters the name and the size into the dictionary (if it is not already there) and also enters the word containing the next available dictionary entry address. This entry will contain the first symbol in this COMMON block and will be used as the head of the chain of all symbols in this common block. The address of the head of chain is saved by the Loader so that the new set of symbols in the COMMON block may be added to the chain. The larger of the two block sizes is retained as the block size.

When the COMMON block has already been assigned memory locations, the respective lengths are compared. Loading terminates, with an appropriate error message, if the assigned block is smaller. When the assigned block is larger or both are equal, loading continues.

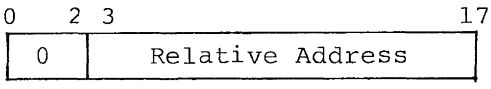
Data Word



15 COMMON Symbol Definition

The data word specifies the relative location of the last symbol loaded (codes 07 and 10) in the last COMMON block (code 14). If the associated COMMON block has been defined (block data), the absolute address of the symbol is calculated (block address plus relative position) and placed in TV location (code 16). When the COMMON block has not been assigned, the relative address is entered into the Loader dictionary and chained to the symbols associated with the COMMON block.

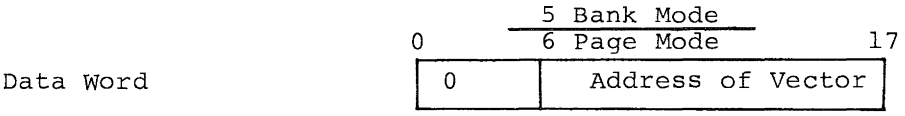
Data Word



Code	Loader Action

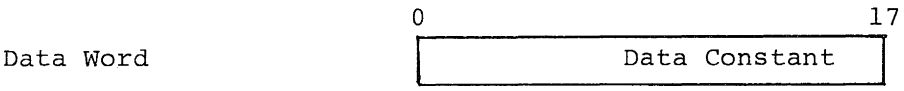
16 COMMON Symbol Reference Definition

The data word contains the unrelocated address of the transfer vector for references to the COMMON symbol named by the last symbol loaded (codes 07 and 10). The symbol definition (code 15) is stored in the relocated (modulo 15 bits) address specified when the associated COMMON block has been assigned (code 14). When the block has not been assigned, the relocated (modulo 15 bits) address is entered into the Loader dictionary along with the relative address (code 15) of the symbol.



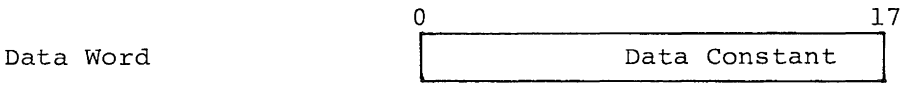
17 Data Initialization Constant - First Word

The data word contains the first machine word of a data initialization constant. It is saved by the Loader for future use (code 22).



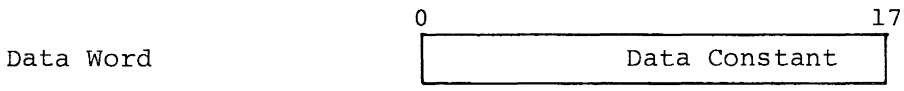
20 Data Initialization Constant - Second Word

The data word contains the second machine word of a data initialization constant. It is saved by the Loader for future use (code 22).



21 Data Initialization Constant - Third Word

The data word contains the third machine word of a data initialization constant. It is saved by the Loader for future use (code 22).

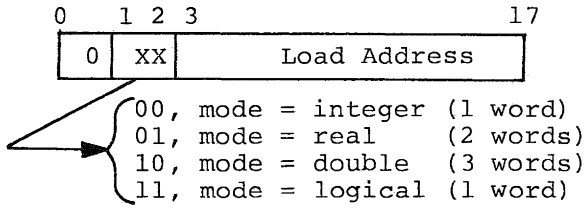


Code	Loader Action

22 Data Initialization Constant Definition

The data word contains the relative load address of the last data initialization constant loaded (codes 17, 20, and 21) and a mode code identifying the constant (real, integer, double, logical). The load address is incremented by the current relocation factor (modulo 15 bits) if the constant initializes a non-COMMON storage element. When the constant initializes a COMMON storage element (indicated by the presence of the block data flag (code 13), the load address is incremented by the address of the last COMMON block loaded (code 14). The constant is stored according to mode and the relocated load address.

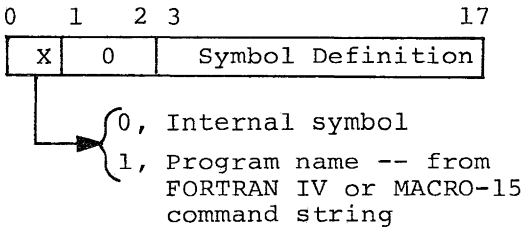
Data Word



23 Program Name or Internal Symbol Definition

The data word contains the unrelocated or absolute address (definition) of the last symbol loaded (codes 07 and 10). The symbol is strictly internal to the program being loaded and is entered conditionally (if a DDT Load) along with its relocated (modulo 15 bits) or absolute address into the DDT symbol dictionary. The program unit name is indicated by bit 0=1 of the data word.

Data Word

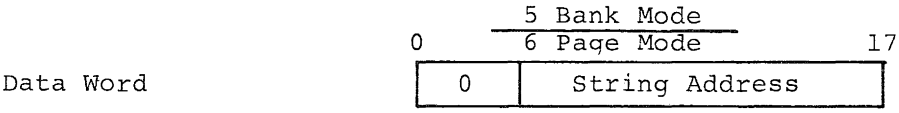


All symbols fall into this category.

Code	Loader Action

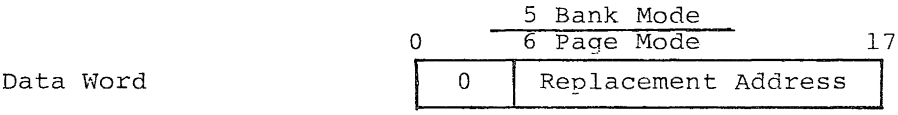
24 String Code - First Half

The data word contains the unrelocated address of a data word whose address portion is to be replaced by another value. The relocated (modulo 15 bits) address is saved by the Loader for future use (code 25).



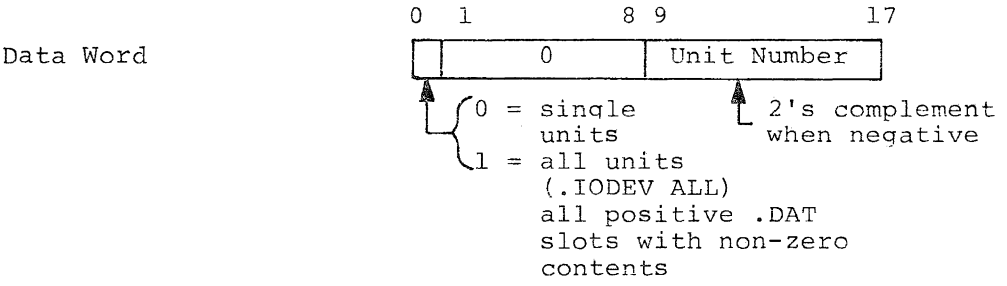
25 String Code - Second Half

The data word contains an unrelocated address. The address portion of the data word specified by the first half-string code (code 24) is replaced with this address (relocated modulo 12 bits (page) or 13 bits (bank)).



26 Input/Output Device Routine Request

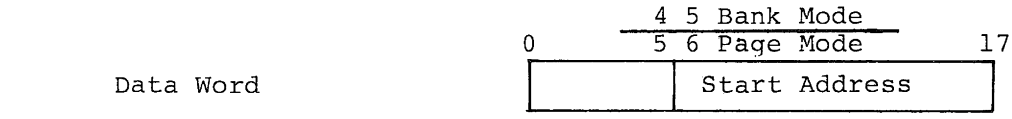
The data word specifies the unit number (.DAT slot number) associated with a device level I/O routine. The Loader defers loading of any I/O routines until all other subprogram loading has been completed; when subprogram loading is complete, the system library is searched for all requested I/O device routines not already residing in memory (see Operating Procedures). The I/O routines are then loaded.



Code	Loader Action

27 End of Program Unit

This information unit is the last unit of a program unit. The data word contains the unrelocated or absolute start execution address of the program. The relocated (modulo 15 bits) or absolute start address is entered into the system communication tables to be used when control is given to the user. Only the first start address encountered is entered into the communication tables. (It is assumed that the first program unit specified in the command string is the main program.) The first address of the main program is used if the .END pseudo-op did not have a start address. When loading from either the system or external libraries, the end unit causes the Loader to examine the next line buffer for the end-of-file (EOF) condition. When the EOF for the external library is obtained, the Loader automatically begins searching the system library to resolve any remaining globals. Upon encountering the EOF of the system library, the Loader announces any unresolved global names. When loading is complete, control goes to the user program, DDT, or to the teleprinter handler in the Monitor as a function of the load command (GLOAD, LOAD, DDT, or DDTNS) (see Operating Procedures).



31 Enable Bank Relocation

This code is output by the MACRO-15 Assembler in response to the .EBREL pseudo-op. The Loader will relocate all Ø3 coded data words using 13 bit addressing. The associated data word is unused.

32 Disable Bank Relocation

This code is output by the MACRO-15 Assembler in response to the .DBREL pseudo-op. The Loader will relocate all Ø3 coded data words using 12 bit addressing. The associated data word is unused.

NOTE

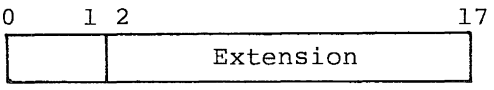
Loader codes 31 and 32 do not affect the execution of the relocated code but merely the size of the address field. Also, these

codes are recognized only when the Loader is operating in Page Mode (PAGE ON or BANK OFF Keyboard Commands). Program units which use these codes are not allowed to overlap memory page bounds or be larger than 4K. These codes are intended for use with VT-15 Display programs and should be used with caution. The instructions EBA (enter bank addressing) and DBA (disable bank addressing) should be with code relocated via the .EBREL and .DBREL pseudo-ops.

33 Source File Extension

This code is output by the MACRO-15 Assembler when operating under the DOS-15 Monitor. The data word contains the extension (in radix 50 format) of the source file which produced this binary. This information is output at load time by selecting the P option in the loader's command string (see paragraph 3.3.1).

Data Word



CHAPTER 3

OPERATING PROCEDURES

3.1 .DAT SLOT ASSIGNMENTS

Prior to calling the Loader, the user should perform all required device and UIC (in the case of DOS-15 systems) assignments for both the Loader and the program to be loaded. Space can be saved during loading if the same version of a handler is used both for the Loader and for the program to be loaded.

All programs named in the command string must reside on the device (and UIC for DOS-15 systems) associated with .DAT slot -4. At least one program must be loaded from this device. The system library (.LIBR BIN) must be assigned to .DAT slot -1. If a user-created library is to be used, it must be assigned to DAT slot -5, and be named .LIBR5 BIN. If no user library is required, .DAT Slot -5 must be assigned to NON; otherwise IOPS 13 (file not found) errors will occur.

3.2 CALLING THE LOADER

The Loader may be called using any of four commands, depending on the user's requirements, as shown:

Command	Meaning
LOAD↵	Load and Halt. Program execution is initiated by typing CTRL P.
GLOAD↵	Load and Go. Program execution begins automatically.
DDT[1]↵	Load user programs along with DDT. When loading is complete, control is given to DDT.
DDTNS[1]↵	Load user programs with DDT but do not build DDT symbol table. This provides more free core but limits debugging to octal numbers.

[1]Refer to the DDT Utility Program Manual (DEC-15-YW2A-DN1).

Type the desired command immediately to the right of the Monitor's $ as follows:

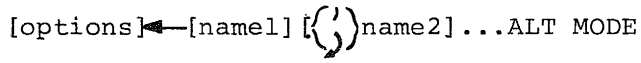
```
    $LOAD↵
or
    $GLOAD↵
or
    $DDT ↵
or
    $DDTNS↵
```

When loaded, the Loader identifies itself with one of the following messages, depending upon the addressing mode (Bank or Page).

Page Mode	Bank Mode
LOADER Vnn	BLOADER Vnn
>	>

3.3 COMMAND STRING

The Command String must be typed immediately to the right of the Loader's prompting symbol (>) in the format shown below:



3.3.1 Option Switches

Three option switches may be selected to obtain loader map output on the teleprinter. If no options are selected, no map is output and loading time is decreased. The switches are as follows:

- P - Type program names and addresses
- G - Type GLOBL symbols and addresses
- C - Type COMMON block names and first address (.XX is the name for BLANK COMMON).

The option switches may be typed in any order and can optionally be separated by commas. In typing out the memory map, the Loader first types the option switch character, followed by the name or symbol and source file extension, followed by the address.

For example:

```
P NAME1 023     37602
P NAME2 002     31547
G .DA   036     27632
G BCDIO 001     27203
G FIOPS 022     26477
C .XX   007     26000
```

3.3.2 Program Names

Program names are standard six character (maximum) file names of the programs to be loaded. The Loader assumes that all programs have a BIN extension. The name of the main program (i.e., the program which is to obtain control first after loading) must be typed first. Alternatively, all programs to be named in a command string could be combined into a library file under the name of the main program.

The name of the main program is followed by the names of all required subprograms which are not to be loaded from the system or user library. Subprogram names should be typed in order of program size, largest first and smallest last, to obtain optimum core utilization. The program names must be separated either by commas or by Carriage RETURNS. If program input is from a non-directoried device, program names are ignored and need not be typed. Simply type n-1 commas or Carriage RETURNs to load n programs.

3.3.3 ALT MODE

An ALT MODE is the only legal command string terminator for the Loader. Once typed, program loading begins.

3.3.4 Command String Errors

Syntactical errors in command strings (e.g., omitting the back arrow (←)) cause the Loader to restart. Typing errors which occur prior to typing a Carriage RETURN or ALT MODE can be deleted through the use of the RUBOUT (delete character) and CTRL U (delete line) teleprinter editing features. The Loader can be restarted at any time prior to typing the ALT MODE terminator by means of the CTRL P command. Command string errors observed after an ALT MODE has been typed are unrecoverable and the user must return to the Monitor (via CTRL C) and reload the Loader.

3.4 OPERATION

Upon receipt of the command string, the Loader consecutively loads all explicitly named programs and builds a symbol table consisting of external (global) symbols, COMMON block names and COMMON blocks. Once all named programs have been loaded, a search is begun to resolve unsatisfied subroutine requests contained in the synbol table. This is accomplished by searching for the various subroutines and device handlers in the IOS[1] user library (if present), and the system library, in that order, as many times as required to resolve the reference. If a complete pass is made with no new resolution, the Loader then tries first to match the references to COMMON blocks and then to COMMON names. Any remaining unresolved references cause a .LOAD 3 error and loading terminates. Unresolved symbols are typed at the end of a loader map (when a map is requested) and have a load address of Ø. The Loader's search normally terminates as soon as all global references are resolved. Programs are loaded from the top of core down, starting with the extra 4K page, when available[2], and continuing into the next bank (just below the system bootstrap), as shown in Figure 3-1. The first 20_8 locations in each page (Page Mode ON) or in each bank (Page Mode OFF) are not loaded. When there is no extra 4K page, loading begins at the top highest bank just below the system bootstrap.

When loading is complete, the Loader resets the free core pointers .SCOM+3 and .SCOM+2 (absolute locations 103 and 102) to indicate that the area occupied by the Loader and its handlers, if they are not used by the loaded program, is now free core. The Loader then passes control either to the user's main program or to DDT in one of several ways as follows:

a. If the Loader was called using the LOAD command, the Loader types ↑S and waits for the user to type CTRL S to start his program.

b. If GLOAD was used to call the Loader, the user's main program is automatically started.

c. If either DDT or DDTNS was used to call the Loader, control is given to DDT (refer to the DDT Utility Program manual (DEC-15-YWZA-DN1) for further information).

---

[1]IOS is the I/O Service UIC in which I/O handlers reside in DOS-15 systems.

[2]The setting of the Monitor's X4K ON OFF Keyboard Command controls the Loader's ability to utilize an extra memory page.

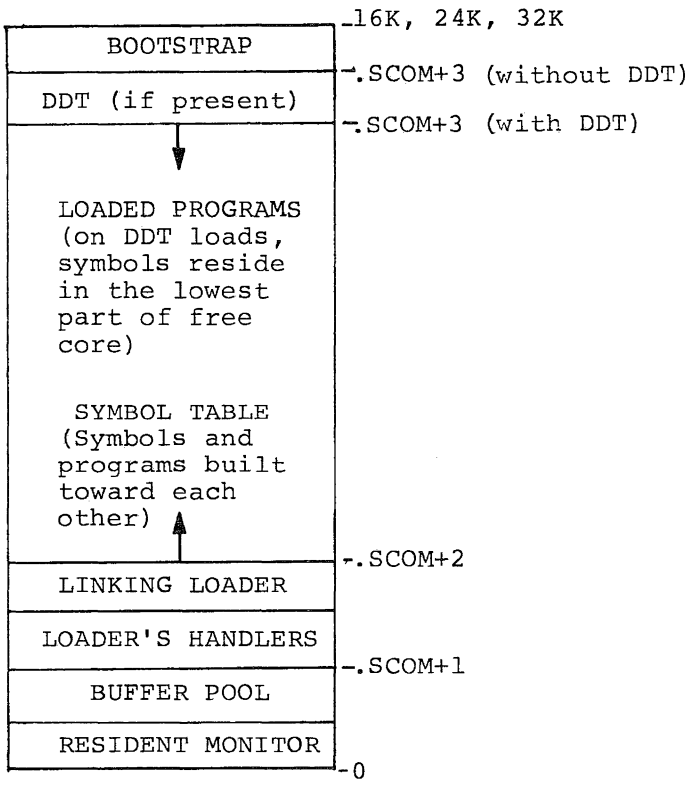


Figure 3-1

Linking Loader Core Map

3.5 ERROR CONDITIONS

The following error codes are output by both the Linking Loader and the System Loader. When output by the Linking Loader, the errors are identified as shown below. When output by the System Loader, the errors are identified as ".SYSLD n" instead of ".LOAD n".

Error	Explanation
.LOAD 1	Memory overflow - the Loader's symbol table and the user's program have overlapped. At this point the Loader memory map will show the addresses of all programs loaded successfully before the overflow. Increased use of COMMON storage may allow the program to be loaded as COMMON can overlay the Loader and its symbol table, since it is not loaded into until run time.
.LOAD 2	Input data error - parity error, checksum error, illegal data code, or buffer overflow (input line bigger than Loader's buffer).
.LOAD 3	Unresolved Globals - any programs or subroutines required but not found, whether called explicitly or implicitly, are indicated in the memory map with an address of 00000. If any of the entries in the memory map has a 00000 address, loading was not successful; the cause of trouble should be remedied and the procedure repeated.
.LOAD 4	Illegal .DAT slot request - the .DAT slot requested was: a. Out of range of legal .DAT slot numbers; b. Zero; c. Unassigned; that is, was not set up at System Generation Time or was not set up by an ASSIGN command.
.LOAD 5	Program segment greater than 4K - the program segment being loaded in Page Mode exceeds a Page Bound.

APPENDIX A

PROGRAMMING NOTES

1. Recommended practice for memory overflow (.LOAD 1) errors --
   Apart from the obvious techniques of segmenting large programs and linking them via .GLOBL's, use of the CHAIN and EXECUTE programs will often allow loading of programs which overflow when using the Linking Loader. Since CHAIN creates XCT files on storage external to memory, space which would have been used by the Linking Loader is made available to the user during the chaining process.

2. Block data subprograms must be explicitly added after the main user program; that is, the name of the block data subprogram must be typed after that of the main program in the Loader command string if it is a separate file.

3. For ADVANCED Monitor and Background/Foreground Users, the recommended assignment for .DAT slot -1 (System Library: .LIBR BIN) in an 8K system is DTC. for DECtape systems and DKC. for disk systems. Note that care must be taken to assign the same handler to both .DAT slots -1 and -4 (user program input) to avoid possible incorrect linkage to one handler interrupt service when the initial I/O call was made to another handler.

   The assignment of the same handler to both the Linking Loader and user .DAT slots prevents the unnecessary loading of extra handlers which only take up more core. For example, if the program being loaded uses .DAT slots 1 and 2 for its I/O, a core-saving technique is to assign a common DECtape handler to .DAT slots 1, 2, -4, -1 and -5 if a user library .LIBR5 BIN exists.

4. .INIT's to negative .DAT slots --
   An .INIT cannot be done to .DAT slots -4, -5, -1 and -7 since the Linking Loader uses these slots and clears them after loading.

APPENDIX B

TERMS AND DEFINITIONS

Term	Definition
Loadable Program Unit	A main program, subprogram, or block data subprogram.
Transfer Vector	A core location containing the address of a subprogram or an entity in COMMON. All references to subprograms and entities in COMMON are indirect.
Internal Global Symbol	A symbol defined in the current program unit and accessible to all programs.
External Global Symbol	A symbol which is referenced in the current program unit and defined in another.
Unresolved Global Symbol	An external global symbol reference which has not yet been resolved by replacement with an internal global symbol definition.
Relocation Factor	The amount added to relative addresses to form absolute addresses; initially, the first loadable core location. The relocation factor for programs following the first program unit is the next available load address.
Radix 50_8 Format	A method of symbol concatenation[1] utilizing 50_8 characters as a "number set", each with a unique value between and including 0 to 47_8. The symbol (number) is converted using standard base conversion methods (see Appendix C).

[1] i.e., linking together

APPENDIX C

SYMBOL CONCATENATION[1] - RADIX 50_8 FORMAT

Radix 50_8 is a technique used by the MACRO Assembler and the FORTRAN IV Compiler to condense the binary representation of symbolic names in symbol tables. Three characters, plus two symbol classification bits, are contained in each 18-bit word. A symbol is defined as a string of one to six characters, i.e.,

$$C_1C_2C_3C_4C_5C_6$$

where any of the possible six characters (C_1 through C_6) can be defined as:

Character	6-bit octal code
Space	00
A	01
↓	↓
Z	32
%	33
.	34
0	35
↓	↓
9	46
#	47

The characters which make up a symbol are linked together in the following manner:

Word 1 $\quad ((C_1 * 50_8)+C_2)50_8+C_3$

Word 2 $\quad ((C_4 * 50_8)+C_5)50_8+C_6$

For example, the symbol SYMNAM would be entered in the Loader's symbol table as:

Word 1 $\quad ((23_8 * 50_8)+31_8)50_8+15_8 = 475265$[2]

Word 2 $\quad ((16_8 * 50_8)+1)50_8+15_8 = 053665$

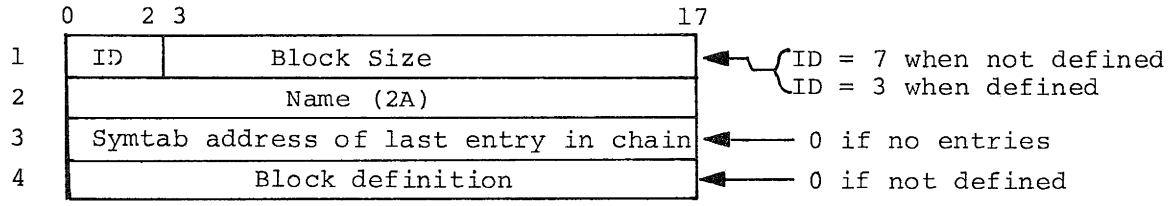
[1] i.e., linking together

[2] The sign bit of WORD1 is set to 1 to indicate that this symbol consists of more than 3 characters and that the WORD 2 is necessary.

APPENDIX D

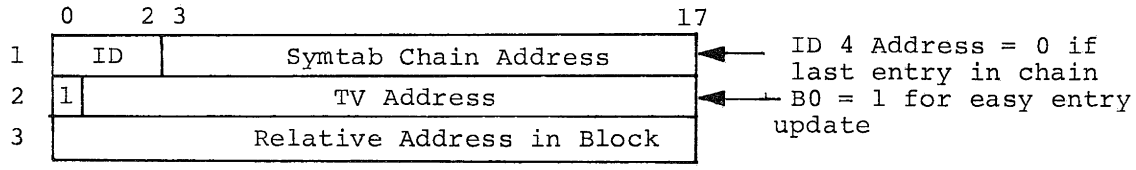
LOADER SYMBOL TABLE

COMMON BLOCK NAME



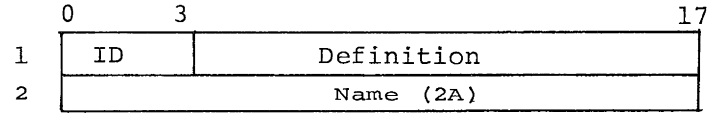
"Name" may require 2 words

COMMON NAME



If associated COMMON block was defined when code 14 is encountered, no entry is needed in the symbol table.

UNRESOLVED OR INTERNAL GLOBAL

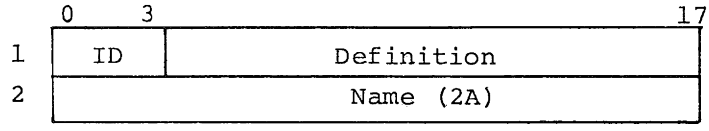


Definition (Unresolved) = Absolute Address of last TV in chain
Definition (Internal) = Absolute address of Symbol

"Name" may require 2 words.

Unresolved ID = 1
Internal ID = 5

INTERNAL NAMES



"Name" may require 2 words

ID=0 (If Program Name ID = 7) Only entered into the symbol table during DDT loads

HOW TO OBTAIN SOFTWARE INFORMATION

Announcements for new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters.

Digital Software News for the PDP-8 & PDP-12
Digital Software News for the PDP-11
Digital Software News for the PDP-9/15 Family

These newsletters contain information applicable to software available from Digital's Program Library, Articles in Digital Software News update the cumulative Software Performance Summary which is contained in each basic kit of system software for new computers. To assure that the monthly Digital Software News is sent to the appropriate software contact at your installation, please check with the Software Specialist or Sales Engineer at your nearest Digital office.

Questions or problems concerning Digital's Software should be reported to the Software Specialist. In cases where no Software Specialist is available, please send a Software Performance Report form with details of the problem to:

Software Information Service
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

These forms which are provided in the software kit should be fully filled out and accompanied by teletype output as well as listings or tapes of the user program to facilitate a complete investigation. An answer will be sent to the individual and appropriate topics of general interest will be printed in the newsletter.

Orders for new and revised software and manuals, additional Software Performance Report forms, and software price lists should be directed to the nearest Digital Field office or representative. U.S.A. customers may order directly from the Program Library in Maynard. When ordering, include the code number and a brief description of the software requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability and readability.

Did you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. _____ Date: _____

Name: _____ Organization: _____

Street: _____ Department: _____

City: _____ State: _____ Zip or Country _____

— — — — — — — — — — — — — — — — — — Fold Here — — — — — — — — — — — — — — — — — —

— — — — — — — — — — — — Do Not Tear - Fold Here and Staple — — — — — — — — — — —

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754