Publication Number 29-231 R11

16· BIT
LOADER DESCRIPTIONS
MANUAL

INTERDATA®
A UNIT OF
PERKIN-ELMER DATA SYSTEMS
2 Crescent Place, Oceanport, New Jersey 07757• (201) 229-4040

May 1977

PAGE REVISION STATUS SHEET

PUBLICATION NUMBER 29-231

TITLE 16-Bit Loader Descriptions Manual

REVISION R11 DATE May 1977

PAGE	REV.	DATE	PAGE	REV.	DATE	PAGE	REV.	DATE
Title Page	R10	2/77	A4-1/ A4-2	R09	5/75			
i/ii	R10	2/77	A5-1	R10	2/77			
iii	R09	5/75	A5-2	R09	5/75			
iv	R10	2/77	A6-1	R09	5/75			
1	R09	5/75	A6-2	R09	5/75			
2	R09	5/75						
3	R09	5/75	A7-1/ A7-2	R09	5/75			
4	R09	5/75						
5	R09	5/75						
6	R09	5/75	I-1	R10	2/77			
7	R09	5/75	I-2	R10	2/77			
8	R09	5/75						
9	R09	5/75						
10	R09	5/75						
11	R09	5/75						
12	R09	5/75						
13	R09	5/75						
14	R09	5/75						
15	R09	5/75						
16	R09	5/75						
17	R09	5/75						
18	R09	5/75						
19	R09	5/75						
20	R09	5/75						
21	R09	5/75						
22	R09	5/75						
23	R09	5/75						
24	R09	5/75						
25/26	R11	5/77						
A1-1	R09	5/75						
A1-2	R09	5/75						
A2-1/ A2-2	R09	5/75						
A3-1	R09	5/75						
A3-2	R11	5/77						

FOREWORD

This manual is designed as a user reference to the INTERDATA 16-Bit Loader family. Chapter 1 is an introduction to the various loaders. The Standard Loader Format is detailed in Chapter 2. Chapter 3 describes the various Stand Alone loaders. The OS Library Loader is detailed in Chapter 4 and the OS Random Access Bootstrap (07-046) is described in Chapter 5. Seven appendixes describe in various operations, commands, and procedures in summary form. An Index is provided at the back of the manual for reference conveniences.

16-BIT
LOADER DESCRIPTIONS

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

APPENDICES

ILLUSTRATIONS

TABLES

16-BIT
LOADER DESCRIPTION

1. INTRODUCTION

1.1 Types of Loaders

The purpose of this manual is to describe to the INTERDATA user, the various loaders that are available, and to aid the user in selecting and using each loader program. There are four main classes of loaders:

1.1.1 50 Sequence Loaders. TYPES: Model 3, 50 Sequence and Model 4 and up, 50 Sequence. A memory resident loader that loads core-image data.

1.1.2 Stand-Alone Loaders. These loaders load object programs output by the INTERDATA assembler, and do not require OS support. TYPES: RELOCATING LOADER, GENERAL LOADER.

1.1.3 OS Loaders. This class loads assembly language and FORTRAN programs under OS control.

TYPES: BOSS RESIDENT LOADER
DOS RESIDENT LOADER
RTOS RESIDENT LOADER
OS LIBRARY LOADER
RTOS TASK ESTABLISHER (TET)

1.1.4 Bootstrap Loaders. These programs act as front-end loaders for INTERDATA Software programs and loaders thus making them self-loading.

TYPES: FAST FORMAT LOADER
SELF RELOCATING LOADER
BULK STORAGE BOOTSTRAP LOADER

1.2 Loader Terminology

The following list is a glossary of terms used in this manual, describing the loader features and program characteristics.

ABSOLUTE PROGRAM	A program assembled to occupy fixed locations in memory.
RELOCATABLE PROGRAM	A program that can be loaded at a desired memory location specified at load time.
BIAS	The address specifying where a relocatable program is to be loaded.
LABEL	A six character name that identifies a program.
STANDARD LOADER FORMAT	An encoding scheme which is used to represent object programs.
ZONED TAPE	A standard loader format tape that allows binary characters to be punched on the Teletype.
EXTRN	A symbolic reference to an external program or value linkable at load time.
ENTRY	A symbolic definition which allows references by other programs to be resolved at load time.

COMMON	An area set aside by a loader at load time for reference by a number of programs.
UBOT	User bottom of core - UBOT is the first (lowest address) memory location available for loading user program in an OS environment.
CTOP	Top of Core - The highest address of physical memory.
RANDOM ACCESS STORAGE	Disc or Drum

1.3 Program Formats

INTERDATA programs are supplied in various formats and on different media. A part number suffix is used to identify the format and media of the program. The suffix used for this purpose is the manufacturing variation denoted by an M followed by a two digit number.

The format for the M field and its meaning for Software is:

M x y

where x identifies the media selection (i.e., paper tape, mag tape, cassette, etc.) and y identifies object or source and the format.

Meaning of x		Meaning of y
Paper tape	1	1 Object program standard format 32 bit Processor
Cassette	2	4 Memory image
Mag tape (800)	3	6 Object program standard format 16 bit Processor
Cards	4	7 Object non-standard format
Disc (2.5 MB)	5	8 Object established task
Disc (5 MB)	6	9 Source program

The above numbers refer to the physical program placed on an approved media for INTERDATA software.

The following loaders may be used on the media specified:

50 SEQ	Paper tape, Magnetic tape and Cassette
Relocating and General Loader	Paper tape, Magnetic tape and Cassette
OS Loader	Paper tape, Magnetic tape, Cassette and Disk
Fast-Format Boot Loader	Paper Tape, Magnetic tape and Cassette
Self-Relocating Loader	Paper tape, Magnetic tape and Cassette
Random Access Bootstrap	Disc or Drum only

REL Loader	06-024R01 or greater
General Loader	06-025R01 or greater
OS Library Loader	03-030
BOSS Resident Loader	03-019
DOS Resident Loader	03-022
RTOS Task Establisher Task	03-042

Refer to Section 2 for a complete definition of standard loader format. All of the above loaders load either zoned or non-zoned tapes.

Zoned tapes use only 4 bits of each tape frame for data with the other 4 bits assigned a "zone" code to make the entire frame a non-printing ASCII code. This format is used for punching tapes on an ASR Teletype tape punch; all INTERDATA programs produce this format when punching on a Teletype device. The non-zoned tapes use all 8-bits of each tape frame for data, and these tapes are, therefore, half as long as zoned tapes for an identical program.

Non-zoned paper tapes use an X'F0' as the first character of each record, so that the loaders can identify the two formats. The Basic Assembler, Hex-Debug, and the corresponding OS programs produce non-zoned tapes when punching to a High Speed Paper Tape Punch. This tape format results in less punching time, shorter tapes, and faster loading than with zoned tapes.

Each of these loaders provide a varying degree of loading capability. A brief description, in ascending order, follows:

RELOCATING LOADER	Loads absolute or relocatable programs and handles forward referencing. No external symbolic referencing is allowed.
BOSS and DOS RESIDENT LOADER	Same features as the RELOCATING loaders except programs are loaded for execution under the OS. For details on features and operation, see the BOSS or DOS Reference Manual.
GENERAL LOADER	Includes features of the RELOCATING loader and resolves symbolic references/definition linkage (EXTRN/ENTRY) and processes program labels.
OS LIBRARY LOADER	Includes features of the GENERAL loader and allows selective Editing of program libraries, building of load modules and overlays, and handling of common.
RTOS TASK ESTABLISHER	Includes most of the features of the OS Library Loader but is specifically designed to establish tasks for RTOS. The established tasks produced by TET are loadable only by the RTOS loader which is part of RTOS. For details on features and operation, see the RTOS Reference Manual.

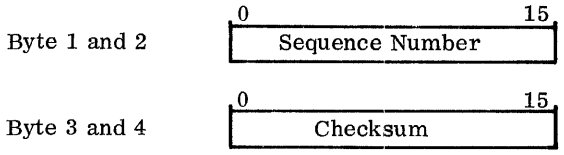
Table 1 is a summary of loaders and their features.

TABLE 1. SUMMARY OF LOADER FEATURES

	REL Loader	General Loader	OS Library Loader	BOSS Resident Loader	DOS Resident Loader	RTOS Task Establisher Task
ABS Programs	Yes	Yes	Yes	Yes	Yes	No
REL Programs	Yes	Yes	Yes	Yes	Yes	Yes
Program Labels	Ignored	Typed Out	Legal	Ignored	Ignored	Legal
ENTRYs	Ignored	Legal	Legal	Ignored	Ignored	Legal
EXTRNs	Illegal	Legal	Legal	Illegal	Illegal	Legal
Common Usage	Illegal	Illegal	Legal	Illegal	Illegal	Legal
Error Handling	Halt	Typed Out	Typed Out	Typed Out	Typed Out	Typed Out
OUT Mode	No	No	Yes	No	No	Yes
MAP Facility	No	No	Yes	No	No	Yes
Library Handling	No	No	Yes	No	No	Yes
Library Building	No	No	Yes	No	No	No
Overlay Building	No	No	R02 or Higher	No	No	Yes
Automatic Post-Load Transfer	Yes	Yes	No	No	No	No
RTOS Task Building	No	No	No	No	No	Yes
System Requirements	1KB	1.5KB	5KB+DOS,BOSS, RTOS, or OS/16-MT	BOSS	DOS	7KB+DOS, BOSS, RTOS, or OS/16-MT

2. INTERDATA STANDARD LOADER FORMAT

Standard loader format binary object tapes are divided into records separated by interrecord gaps. Each record contains 108 bytes (with 108 zone codes for zoned formats) of information. The first four bytes of data are organized as follows:



The sequence numbers are sequential negative integers -1, -2, -3, etc., represented in two's complement form. The first record in a program must have sequence number -1. Subsequent records must be in proper order to be loaded.

The checksum is an odd parity Exclusive OR sum of all words in the record, except itself, plus a word of all ONEs.

The remainder of the record is a sequence of items; an item is 4-bits or a half-byte. There are two types of items; control items and data items. There are 20 basic control items, each of which is followed by a certain number (which might be zero) of data items. The control items, and their meanings, are listed in Table 2. The RTOS Task Establisher (TET) generates additional control items for RTOS use only. These items are listed in Table 2A.

TABLE 2. CONTROL ITEM DEFINITIONS

Control Item	Meaning	Number of Data Items Following
0	Read next record	0
1	End of Program	0
2	Define chain	0
3	Toggle abs/rel mode	0
4	Load transfer address	4
5	Load program address	4
6	Load reference address	4
7	Load definition address	4
8	Data, 2 bytes absolute	4
9	Data, 2 bytes relative	4
A	Data, 4 bytes absolute	8
B	Data, 4 bytes relative	8
C	Symbol, reference	12
D	Symbol, definition	12
E	Decode Next Item	1
E0	Define Common-block	16
E1	Reference Common-block	16
E2	Common data, 2 bytes absolute	20
E3	Common data, 4 bytes absolute	24
E4	Reset sequence number to -1	0
F	Program Label	12

TABLE 2A. RTOS TASK ESTABLISHER SPECIAL CONTROL ITEMS

CONTROL ITEM	MEANING	NUMBER OF DATA ITEMS FOLLOWING
E5	Task Control Block record indicator, followed by size of the TCBs	2
E6	Index record indicator, followed by number of index records	2
E7	Data record indicator, followed by number of data records	6

Physically, three tape formats are used, all of which contain 108 bytes of data per record with blank leader or inter-record gaps between records. The Loaders use the first non-blank character of each record to identify which format is being used except in the case of magnetic tape or cassette.

Standard Loader Non-Zoned Format: This format is used when punching paper tape on a high speed punch. The first non-blank character of each record contains a X'F0'. This character is followed by 108 frames each containing one data byte. Thus a complete record is 109 frames, of which the first is ignored.

Standard Loader Zoned Format: This format is used when punching object programs on a Teletype punch. A special set of non-printing characters is used, such that only the four low order bits of each frame contain data. (See Table 3.) If one of these special characters is encountered before an X'F0' character is read, the record is assumed to be of this format. Because only 4 bits of each frame are used for data, the 108-byte record is punched in 216 frames of paper tape, of which the upper 4 bits are ignored by the loader input routine.

TABLE 3. TAPE CODES

(ZONED FORMAT)*

Binary		Hex	
Zone	Data	Zone	Data
1001	0000	9	0
1000	0001	8	1
1000	0010	8	2
1000	0011	8	3
1000	0100	8	4
1001	0101	9	5
1001	0110	9	6
1001	0111	9	7
1001	1000	9	8
1001	1001	9	9
1001	1010	9	A
1001	1011	9	B
1001	1100	9	C
1001	1101	9	D
1001	1110	9	E
1001	1111	9	F

* The zoned format is established to allow data that represents the four control TTY characters X'91', X-ON, X'92', TAPE ON, X'92', X-OFF; and X'94', TAPE OFF; to be punched on a teletype punch.

Either format may be read through any paper tape reader.

Format: These formats are used with Magnetic Tape and Cassette (Intertape) devices. They are identical to non-zoned format except that there is no leading X'F0' character. Additional parity checking is accomplished in the Magnetic Tape and INTERDATA hardware to insure a higher degree of data validity in addition to the checksum.

Figure 1 shows a loader record in three formats.

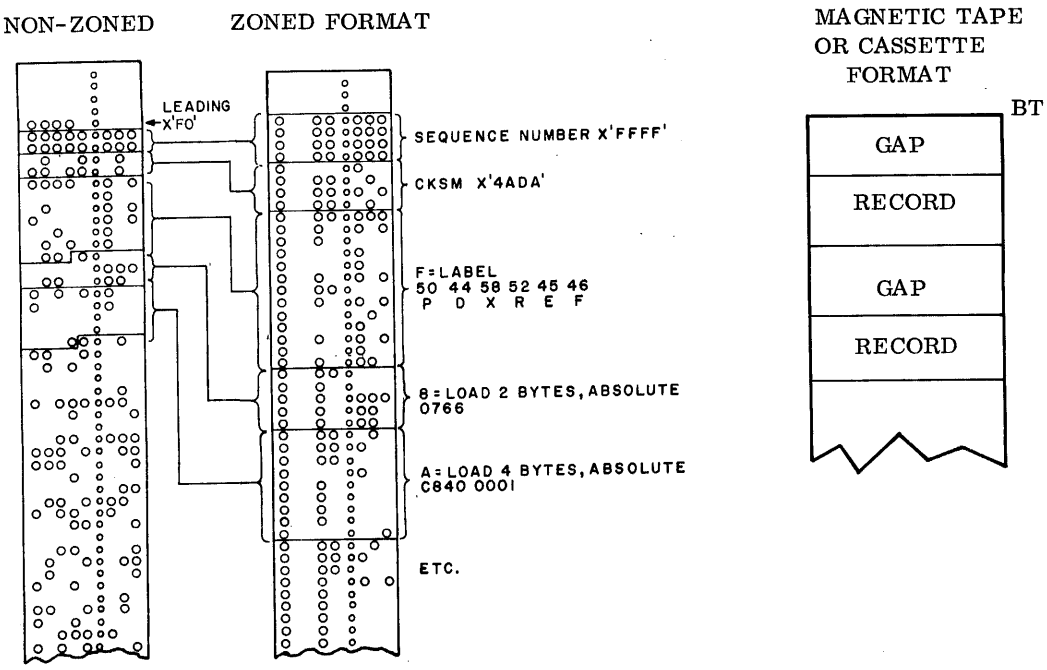


Figure 1. Object Tape Formats

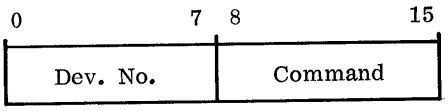
3. THE STAND-ALONE LOADERS RELOCATING LOADER (06-024) GENERAL LOADER (06-025)

3.1 Common Features of the REL and GENERAL Loaders

The stand-alone loaders contain drivers for Teletype, High Speed Paper Tape Reader, Intertape (Cassette) and Magnetic Tape devices. The following description applies to the REL and General Loaders.

1. The input device for loading is defined by the Binary Input Device in the Device Definition Table in low core. A device address of X'n2' is interpreted as a TTY, X'n3' as a High Speed Paper Tape device and X'n5' as a Magnetic Tape or Cassette device.

   This halfword for various devices is shown below.



Examples:		
	Teletype	0294
	High Speed Paper Tape Reader	0399
	High Speed Reader/Punch	1399
	Magnetic Tape	85A1
	Intertape (Cassette)	45A1

2. When reading binary data from paper tape, leader and illegal characters are skipped. Formats are discussed in Section 2.

3. Checksums and sequence numbers are checked after each binary record is read. Appropriate error halts occur when errors are detected.

4. All loaders are provided in relocatable bootstrap form. Specifically, each tape contains a relocating bootstrap sequence followed by the actual loader in normal relocatable object format. When using the bootstrap sequence, the loader is placed into the top of available memory (REL at CTOP-X'400', General at CTOP-X'600'). Once the REL or General Loader is in the top of core, any program can then be loaded to any other point in memory.

5. The first location (ORG) of each loader is the starting location.

6. In the REL loader, Display Panel Indicators 8-15 are used to identify the meaning of loader halts. The light patterns used are:

XX00	normal end
XX01	checksum, parity error, or Device unavailable
XX02	sequence error
XX03	attempt to load over loader
XX04	reference-definition loop
XXFN	Illegal loader item encountered

(N = improper loader item type)

Parity errors detected while reading from Magnetic Tape or Cassette cause the REL Loader, at R03 and above, to backspace and retry the operation 5 times. After the fifth attempt, the record is backspaced and a checksum/parity error pattern is displayed on the Display Panel. Depression of the 'execute' switch causes the record to be re-read. There is no provision for bypassing a record with a parity error. If the device is unavailable (off line), the checksum/parity error pattern is also displayed.

7. The REL and General Loaders, at R03 and above, have the capability of selectively loading programs from Magnetic Tape and Cassette libraries. Programs (separated by file marks) may be specified by entering on switches 0-7 of the Display Panel the number of file marks to be skipped by the Loader. When started, the REL and General Loaders read the Display Panel, skip the specified number of file marks, and begin loading. Care should be taken to avoid inadvertent skipping. For example, when the user is starting the Relocating or General Loaders by addressing NC00 or NA00 in the switches and intends to load the first file on a Cassette or Magnetic Tape, the user must clear (reset) the Switches 0-7 prior to depressing execute. When loading from Paper Tape devices, the Display Panel is ignored.

8. The Loaders transfer immediately to the program loaded, if a transfer address (via an END statement) is on the object tape.

When the REL or General Loader is executed at its starting address (ORG), the Bias value is set to X'80'. This Bias value is used during program loading to adjust any relocatable data values. Note that absolute programs are stored at the absolute location specified for the data. Relocatable programs are stored from the location indicated by Bias upward into memory. After a program has been loaded, the Bias value is adjusted to point to the next available location. To indicate that the load is complete, the General Loader halts with the Wait light on, and with XX00 contained in the Display Register. At this time, the adjusted Bias value is held in Register 0. This register may be displayed on the panel. The REL Loader automatically starts the loaded program if a transfer address was specified; otherwise a similar halt occurs.

If more programs are to be loaded, place the next Paper Tape in the reader or set the Display Panel for Magnetic Tape or Cassette skipping, and with the MODE CONTROL switch in RUN, depress EXECUTE. This procedure starts the General Loader executing at ORG + 26, which is the continue location. The continue operation uses the current value of Bias, and does not reset it to X'80'. Multiple relocatable programs are thus loaded one after another into adjacent areas of memory.

If it is desired to load a relocatable program other than at X'80' in memory, it is necessary to redefine the Bias value. To adjust the Bias pointer, use the following procedure.

1. Change the halfword at ORG + X'A' in the Loader to the desired Bias value.

2. Start the Loader executing at ORG + 8, rather than the normal start or continue location.

Note that the value at ORG + X'A' remains until changed to a new value. The Loader can always be restarted at ORG + 8 which resets the Bias to the value contained in ORG + X'A'.

Forward Reference Definitions

Program object tapes generated by one-pass assemblies or Load Modules from the OS Library Loader involve forward references to symbols which are defined later in the program. The Loader uses a chaining procedure for satisfying any forward references at the time the symbol definition is encountered.

An example of a forward reference in a program is:

```
        OPT     PASS1,PUNCH
        LH      3,SAM
        BR      5
SAM     DC      3
        END
```

In this case, the reference to SAM occurs before SAM is defined. There are several restrictions on the use of forward references during one-pass assemblies, and on the use of symbols which are ENTRYs or EXTRNs for the program to be loaded properly. The restrictions are:

1. Such symbols must not be combined in arithmetic expressions such as:

```
LH  3,SAM+2
```

2. Such symbols must not be used in the R1 or R2 field for an instruction such as:

```
LH  3,2(SAM)
```

3. Such symbols must not be used as operands assembler pseudo-ops such as DO, EQU, END etc.; for example:

```
DO  SAM
```

3.2 General Loader Features

In addition to the capabilities already discussed, the General Loader provides various features not available with the Relocating Loader.

Bias Printout

At the start of every load operation, the General Loader types the current value of the Bias pointer on the Teletype. This printout occurs prior to reading the first record of a new program, and the message is of the form

BIAS = XXXX

where the XXXX represents the current Bias value in hexadecimal form.

Messages

Other messages which are typed on the Teletype are as indicated in Table 4.

ENTRY/EXTRN Handling

Programs generated by the assembler can use ENTRYs or EXTRNs to achieve cross referencing and linkage with external programs. In this case, the object tape for these programs contains the symbolic names declared as ENTRYs or EXTRNs. The General Loader uses a symbol table to remember these names when a program is loaded. This symbol table builds downward in memory from the origin (ORG) of the General Loader. Each entry in the Loader symbol table requires 8 bytes of memory.

Since the Loader symbol table is building downward into memory, and the programs being loaded are building upward into memory, the Loader checks to see that the loading program does not overwrite the symbol table. If the loading program requires data stored above the current bottom of the symbol table, a MEMORY FULL message is generated and the Loader halts.

When the General Loader is executed at its start location (ORG) or its BIAS redefinition location (ORG + 8), the symbol table is cleared of all names. Executing the General Loader at its continue location (ORG + 26) does not change the state of the symbol table.

TABLE 4. GENERAL LOADER ERROR MESSAGES

MESSAGE	MEANING
CKSM ERR	A checksum error was detected after reading the previous record. Reposition the tape to the beginning of the record and depress EXECUTE to reread the record. Magnetic Tape and Cassette parity errors cause the record to be backspaced. Depressing EXECUTE will reread the record.
SEQ-ERR	A sequence number error was detected after reading the previous error. Reposition the Paper Tape to the proper record and depress EXECUTE to try again. This error usually occurs when the tape is improperly positioned following a checksum error.
MEM-FULL	This message is caused by a conflict between the General Loader and the loading program. The program being loaded has not been loaded to conclusion. The alternatives are: A. Load fewer programs B. Make absolute tapes of the programs to be loaded and then use the REL Loader which requires less memory C. Eliminate some EXTRNs and ENTRYs to reduce size of symbol table Note that the General Loader cannot load programs above itself in memory.
NORMAL END	This case occurs when a program has successfully loaded and no END transfer address has been specified or if undefined external references remain. All undefined external references are listed on the Teletype preceded by a U prior to printing the NORMAL END message. If a transfer address is specified and no undefined symbols remain, the Loader transfers directly to the address specified, and no NORMAL END message occurs.
LOAD ERR	This message results if an illegal control item is detected during load. Depress EXECUTE to ignore the control item and attempt to proceed with the load. The E control item is explained in Section 2.
REF-LOOP	This message results if an endless forward reference or external reference chain is encountered. It indicates that the input tape was generated incorrectly.

At the end of each program load, the symbol table is scanned for undefined symbols. Any undefined symbols are typed in the form:

U XXXXXX

where XXXXXX is the symbol name. All such undefined names are printed preceding the NORMAL END message. An undefined symbol results from the fact that the symbol was declared and referenced as an EXTRN in some program, and no program yet loaded has declared and defined that same symbol as an ENTRY. As soon as some loading program declares and defines that symbol as an ENTRY, the symbol becomes defined. If more than one program declares and defines a symbol as ENTRY, the message:

M XXXXXX

where XXXXXX is the symbol name, is typed at the time the multiple definition occurs. In this case, the first value defined remains in the symbol table, and the second definition value is ignored.

At the end of each program load, the Loader transfers immediately to the program that has been loaded, only if a transfer address is specified on the tape, and if the symbol table contains no undefined symbols. If any symbols in the table are undefined at the end of a load, those symbols are listed, NORMAL END is printed, and the Loader halts, waiting to load the next program.

Label Handling

Programs generated by the assembler can be labeled through the use of the OPT pseudo-op such as:

```
OPT PASS2, PUNCH, LAB=ABCDEF
```

The program label can be up to 6 characters. The first character must be a letter; subsequent characters can be letters or digits. The object tape, in this case, contains the program label in symbolic form. When the General Loader detects a program label, the label is typed in the form:

```
LABEL = ABCDEF
```

If object tapes which contain labels are loaded by the REL Loader, the label is ignored by the loader.

3.3 Operation of Stand-Alone Loaders

The steps required to load and operate the Stand-Alone Loaders are summarized below.

1. Manually enter the 50 Sequence into memory if it is not already there. Specify the device to be used for loading at location X'78', the Binary Input Device definition. See Appendix 3 for a listing of the 50 Sequences.

2. Place the Loader Tape or Cassette on the input device. Magnetic Tape and Cassette should be in the 'LOAD' position, (LOAD POINT).

3. Enter X'0050' into the display panel switches, select ADRS Mode and depress EXECUTE.

4. Depress INITIALIZE. Select RUN Mode, and depress EXECUTE.

5. If an ASR 33 Teletype is being used as the input device, it is necessary to toggle the reader switch to START, which starts the tape moving. If an ASR 35 Teletype is in use, the Mode switch should be put in KT position and the reader switch should be put in RUN to start the tape. If a High Speed Paper Tape Reader, Magnetic Tape or Intertape is in use, the tape starts under program control.

6. If no input errors occur, the entire tape is read to the end, at which time the Processor halts with the Wait light on, and XX00 contained in the Display Register.

7. If checksum errors are detected during Paper Tape input, the tape stops and the Processor halts with the Wait light on and XX01 contained in the Display Register. When this occurs, reposition the tape to the previous record gap, and depress EXECUTE to reread the record. If the error halt occurs due to the first record on the tape, restart the entire load procedure. If errors occur with Magnetic Tape or Cassette, restart the entire load procedure.

8. Put the Paper Tape to be loaded into the tape reader. If the tape to be loaded is relocatable, and a specific Bias value is required, enter the Bias value into ORG+X'A', and set the starting address to ORG+8. If the tape to be loaded is absolute, or if the current Bias value is satisfactory, set the starting address to ORG. Depress INITIALIZE. Select RUN Mode and depress EXECUTE. If the program to be loaded is on Magnetic Tape or Cassette, place the number of file marks (hexadecimal) for positioning in the switches of the Display Panel and proceed as with Paper Tape. After the specified file marks have been skipped, the loader begins loading immediately without halting.

9. If improper control items are detected during the load, the tape stops, and the Processor halts with the Wait light on and XXFn contained in the Display Register, where n is the bad control item. When this occurs, it must be determined if the right loader is being used. That is, if the object tape involves ENTRYs or EXTRNs or forward references, the General Loader should be used. If Common blocks are called for, the OS Library Loader should be used. If the Loader is appropriate, and the tape is proper, depress EXECUTE to skip the improper data and proceed with the load.

10. If checksum errors are detected during the load, the tape stops and the Processor halts with the Wait light on and XX01 contained in the Display Register. Reposition the tape to the start of the last record read, and depress EXECUTE to reread the record. If parity errors occur with Magnetic Tape or Cassette, the record is automatically backspaced for retrying. The General Loader does not automatically retry but the REL Loader retries five times.

11. When the load is complete, the tape stops. If no transfer address is specified on the tape, the Processor halts with the Wait light on and XX00 contained in the Display Register. If a transfer address is specified, the Relocating Loader transfers directly to the location specified. The General Loader transfers only if the symbol table contains no undefined symbols.

12. If more tapes are to be loaded, return to Step 8 and repeat the process. This loading process is summarized in Figure 2.

3.4 REL and GENERAL Loader Bootstrapping

The General and Relocating Loaders are provided in a relocating bootstrap form. The format of the tapes is illustrated in Figure 3. The tapes consist of three segments: the boot portion in eight-bit format, a fast-format copy of REL Boot Loader, and the actual loader in standard M16 binary object tape format. When the tape is loaded using the eight-bit loader at X'50', the following sequence of events takes place.

1. The eight-bit Loader at X'50' reads another Loader into X'80' to X'CF' and transfers to X'80'.

2. The program at X'80' reads the balance of the eight-bit data into X'DC0' to X'FEA', which includes a Fast Format Loader.

3. An arithmetic checksum on the information from X'DC0' to X'FEA' is then tested. If the checksum is correct, the process continues. If the checksum is not correct, the tape is stopped and the program halts.

4. The Fast Format Loader then loads a special REL Boot Loader from X'80' to X'2D0'.

5. The top of memory is then determined with a search technique, and the REL Boot Loader's Bias is set a fixed distance from the top of core. The REL Loader is placed X'400' from the top of core. The General Loader is placed X'600' from the top.

6. The REL Boot Loader then reads the Loader program, which is in relocatable form, and relocates it into the top portion of core memory.

7. The REL Boot Loader computes checksums on each record, and halts whenever a checksum error is detected. In this case, reposition the tape to the previous record gap and depress EXECUTE to reread the record. If loading from Magnetic Tape or Cassette, rewind the tape and re-start the load process.

8. When the entire tape has been loaded, the Processor halts with the Wait light on.

This sequence requires that the proper 50 Sequence is used, including the Binary Input Device Definition in X'78'. The 50 Sequences are shown in Appendix 3. Appendix 5 includes the construction procedures used in boot-strapping the Relocating and General Loaders and an Execution Map of these loaders as their loading is described in the previous 8 steps.

Since the Loader portion of each tape is a relocatable object tape, it is possible to put the Loaders anywhere in memory. This can be done by using a bootstrap load to get the Relocating or General Loader into the top of memory. The BIAS can then be adjusted and any Loader can then be relocated to any arbitrary point in memory. Once relocated, CLUB can be used to dump an absolute tape of the loader in that location.

CAUTION

WHEN LOADING THE BOOTSTRAP LOADER TAPES, MEMORY FROM 80-2D0 AND DC0-FEA IS USED. ANY PROGRAMS IN THIS AREA OF MEMORY ARE OVER-WRITTEN.

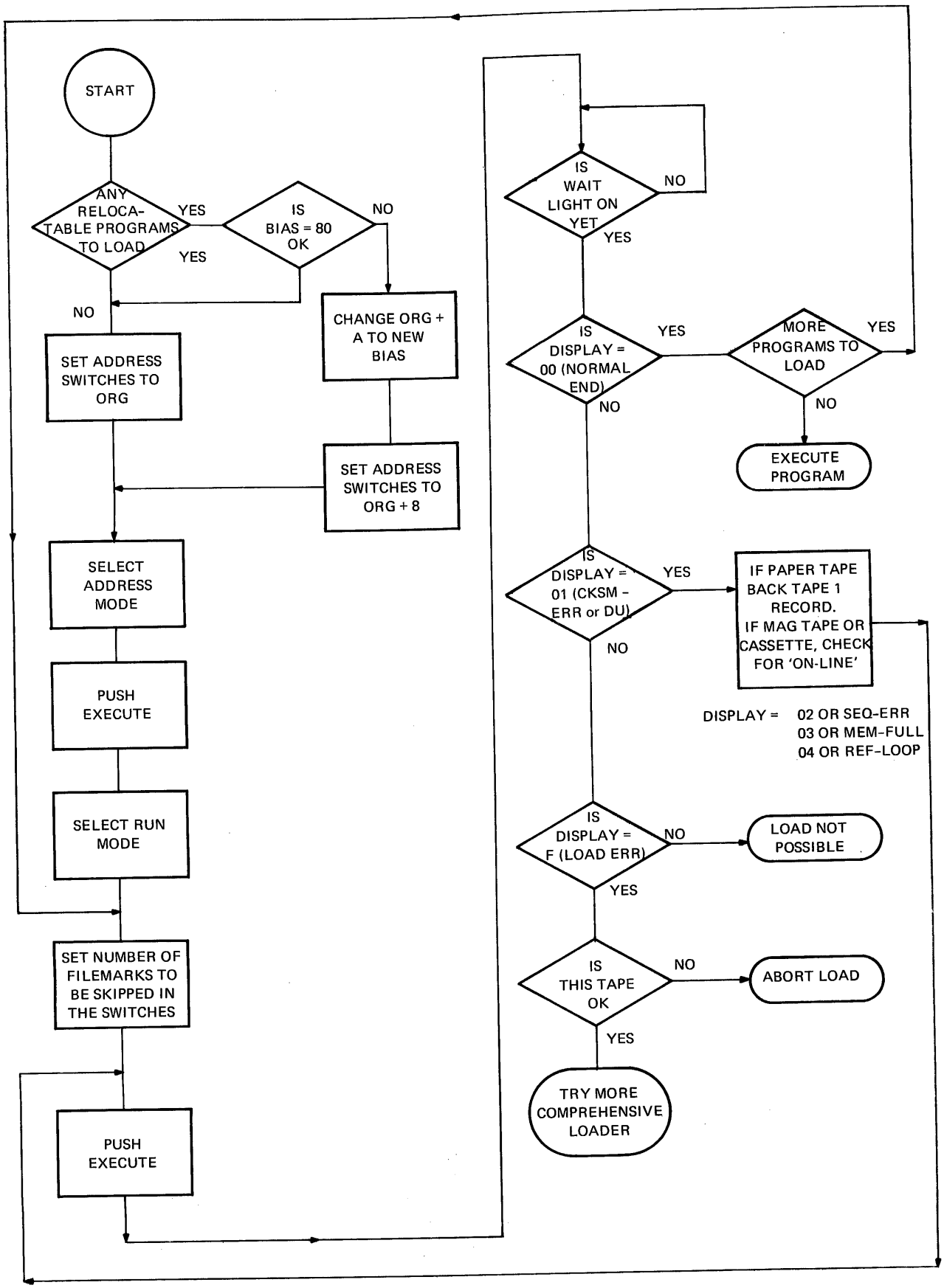


Figure 2. Operating Procedures

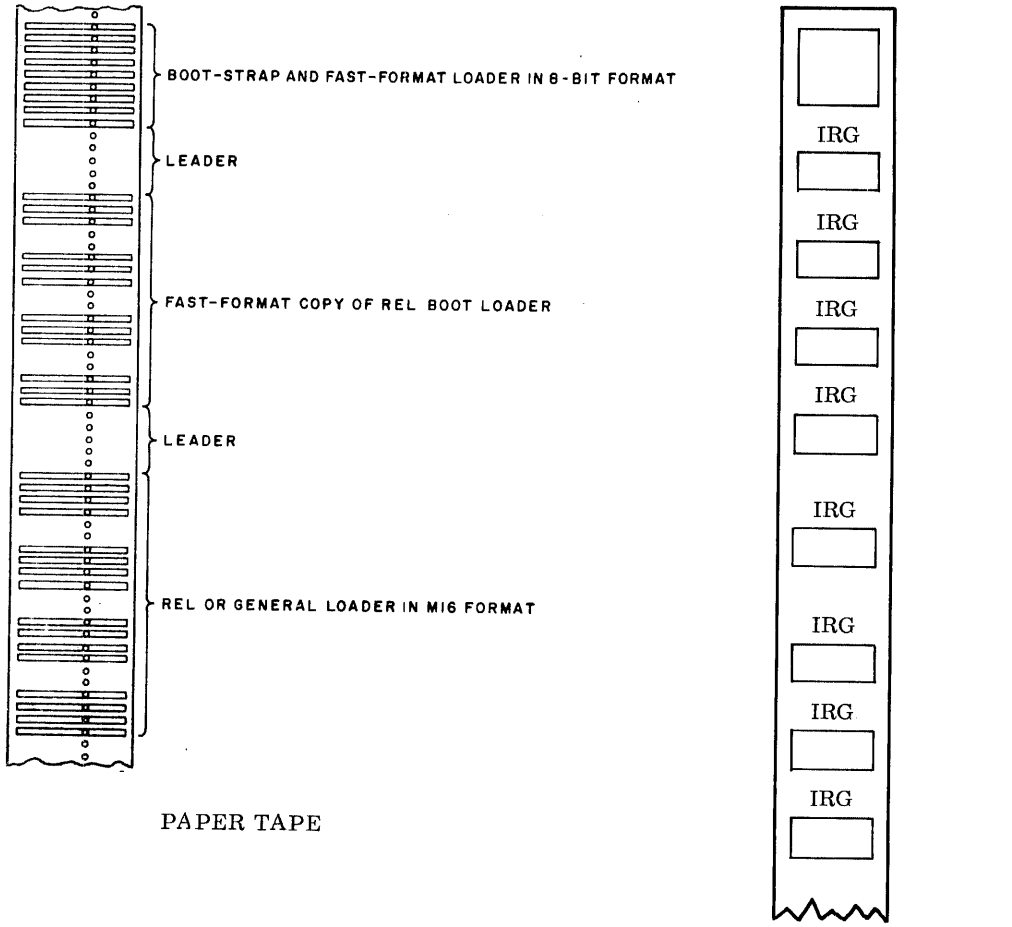


Figure 3. Loader Tape Format

4. FEATURES OF THE OS LIBRARY LOADER (Linking Loader)

NOTE

This document describes LIBRARY LOADERS (Linking Loaders) of revision level R06 and higher. For revision information see Appendix 7. This revision is a major departure from R05 and lower.

This loader functions under the control of an operating system (e.g., BOSS, DOS, RTOS or OS/16-MT) interactively accepting commands and responding with messages to the console operator. Through available operator commands, a file of programs may be created on a bulk storage device. This file may be searched for a particular program, selectively copied onto another I/O unit, or be added to under the operator's control. Automatic Editing is available, allowing the operator with one command to load all library programs required for any one particular job. A group of programs may be pseudo loaded, producing a single absolute load module which may be originated at any address, and then loaded by a Resident Loader, the Relocating Loader, or the General Loader. The modules would contain forward references but no ENTRY or EXTRN symbols. Object programs linked by the OS Library Loader may reference external symbols, define Common Blocks, and contain forward references.

The loader is available in two functional variations, Program Number 03-030F00R06 and Program Number 03-030F01R01, the second of which includes facilities for the generation and maintenance of a file of binary, memory image overlays and Transient Tasks which may be loaded by OS/16-MT.

4.1 Library Loader Operation and Commands

The Library Loader should be loaded by a Resident Loader at the bottom of User Memory. Any I/O devices to be used by the loader should be assigned prior to starting the Loader at its origin. It outputs the message LIBLDR-R06 indicating that it is ready to accept operator commands (which it requests from logical unit 5). Every operator command consists of a word (of which the first two characters are decoded), followed optionally be a space and a hexadecimal operand followed optionally by another space and an alphanumeric argument. A carriage return terminates the command if it is entered via a Teletype. Optional fields which are not used may be ignored. An example and description of each command follows. Following the execution of any command, a prompt character (>) is logged, and another command may be entered. See Appendix 1 for a summary of OS Loader Commands.

BIAS bbbb

This command sets the Loader Bias, and thus the origin of the next program loaded, to bbbb. After loading any program, unless a BIAS command is issued, the Bias is updated to the next available location above the highest address used. When the Loader is initially loaded, the Bias is set to the next location above the Loader. If no Bias value is specified in this command, the Bias is set to the next location above the Loader.

FIND LU xxxxxx

This command causes logical unit LU to be searched forward for a program labeled xxxxxx. The unit is then backspaced one record to the beginning of the found program. The program may now be LOADed or LINKed. If the LU is not a rewindable device, it should be positioned before the first record prior to issuing this command, and backspaced one record after this command completes, before attempting to load the found program. In the event that the requested program is not found, the Find Command will terminate whenever an end-of-file or device end is encountered on the device. In that case, the messages:

EOF

xxxxxx NOT FOUND

are logged on the console device.

LOAD LU xxxxxx

This command should be used to load the first of a group of user programs not related to any previous job. This command initializes the Loader and Loads a program at the current value of Bias. Before the program is loaded, the Loader's symbol table is cleared, and previously defined Common blocks are released. If a program Label is specified in the command, logical unit LU is searched for program xxxxxx. (See FIND.) If no Label is specified, the first program encountered on logical unit LU is loaded. Following a LOAD operation, the logical unit specified is left positioned past the end of the program loaded.

LINK LU xxxxxx

This command Links additional programs needed to complete the module. It must follow a Load command. The symbol table is not cleared, so the loaded program may be linked to any previously loaded program, and may reference previously defined Common blocks. The program is loaded using the current value of Bias. If a program Label is specified in the command, logical unit LU is first searched for program xxxxxx. If no Label is specified the first program encountered on logical unit LU is loaded and linked. Following a LINK operation, the logical unit specified is left positioned past the end of the program loaded.

LF (LINKFILE) LU xxxxxx

This command is the LINK command automatically iterated until an EOF, EOM condition occurs. Encountering a program label of 'ENDVOL' will cause an EOF condition to occur.

EDIT LU xxxxxx

This command is used to search a file, and selectively load those programs needed to satisfy external references (EXTRNs) within programs previously loaded. The criterion for loading during an EDIT operation is a match between the Label of a program in the file and the symbolic name of any EXTRN in the Loader's symbol table. The EDIT command causes the following to occur.

1. If a program Label is specified in the command, logical unit LU is searched for program xxxxxx as described under FIND.

2. The Loader's symbol table is searched for undefined EXTRNs. If there are none, the operation terminates.

3. If there are undefined EXTRNs in the symbol table, logical unit LU is searched forward from its current position for a label matching an undefined EXTRN. If such a Label is found, that program is linked in, and Step 2 is repeated. If an end-of-file or device end is encountered while searching, an appropriate message is logged and the operation terminates. If the Label ENDVOL is read, the EDIT operation is unconditionally terminated. This Label may be placed at the end of each tape of a paper tape program library to prevent reading off the end of the tape.

Following an EDIT operation, the MAP command can be used to determine if any more undefined EXTRNs remain in the symbol table.

MAP LU, Output an address ordered map.

AMAP LU, Output an alphabetically ordered map.

This command outputs a Memory Map to logical unit LU. This may be a map showing the location of programs loaded into core memory, or it may show the locations of programs output in a load module. In either case, the map includes only those programs processed during and after the last LOAD operation. (LINK and EDIT operations add to the map, LOAD begins a new one). The map shows the starting address of each labeled program, the next available address, the location of every ENTRY defined, the starting address of each common block defined, and a list of any undefined EXTRNs. An example of a memory map appears in Figure 4. Under PROGRAMS in the map, the last value shown which has no name is the next available address in memory, which is the current value of Bias. Under COMMON-BLOCKS in the map, Blank Common is identified by the Label //.

If an overlay structure is being built for execution under the Disk Operating System (DOS), the map will always reflect the root segment but only the last overlay segment will appear. Upon entering the overlay command (OV), all entries to programs above the overlay point are removed.

BC nnnn
LC nnnn

These commands must be issued before the LOAD Command if the program being loaded allocates Common. The Loader handles two types of Common. They are Labeled Common, and Blank Common. The chief difference between the two is that Labeled Common may be preloaded with data (FORTRAN BLOCK DATA) while Blank Common may not. The Loader allocates memory for common variables and links common variable references with the associated blocks of core at load time. In order to use core efficiently, the Loader must know in advance, the total length, in bytes, of each kind of Common used. BC sets the maximum length of Blank Common and LC sets the maximum length of Labeled Common to nnnn, where nnnn is the number of bytes to be reserved for common, expressed hexadecimally.

GO

This command transfers control from the OS Library Loader to the transfer address of the loaded programs. If no transfer address is specified, the message CMD-ERR is logged.

```
PROGRAMS:
  0000 OS16MT    2154 .TRAPS    2764 SYSTAB    2B54 TSTPRG
  50DE CRDVR     5274 DCB04     535C LPDVR     5418 DCB62
  545E DSCDVR    5718 DCBC6     583A

ENTRY-POINTS:
  0000 OS16MT    0001 TTYMOD    0034 TCBSZE    0044 DCBSZE
  004E TBSIZE    0077 CLKINT    00C0 SYSGO     02D0 ACTTCB
  020A HR        02F6 INIT      04A4 SCHED     054E EXEC
  05C4 EXEMM1    0608 SIMMM4    0734 SVCIL     083E IOSET
  08B0 IISVC     08EC IOIWAT    0906 BUSY      09DA IOTERM
  0A08 IODONE    0A26 IOEXIT    0A34 IORSAV    0AB6 SUPACT
  0AC0 IOLIST    0AC6 IOPTST    0ACA CYLSIZ    0ACC SECSIZ
  0ACE SCBSY     0ACE SC1BSY    0CAC BTIME     0FE2 CONSH
  11B4 INITSK    11CC LOCT      123C PLINK     18AC CMTKTB
  1904 $CMTSK    1968 CMEOJ     1EA2 OLTKTB    1F00 $OLTSK
  2038 OLEOJ     2154 .TRAPS    2764 TTYOQ     2772 LIOTRM
  279E TCBTAB    27C5 TCBTE     27C6 PRITAB    27D9 PRITBE
  27DA TCBBLK    27DA CMTCB     280E OLTCB     2AE6 TCBTIM
  2B4D TCBE      2B4E DEVTAB    2B54 TSKT01    2BA2 TEST01
  2C16 TSKVRT    2C70 CONVRT    2C9C TSKT02    2CEA TEST02
  2D1E TSKTBB    2D6C TASKBB    2D82 TSKTCC    2DD0 TASKCC
  2DE6 TSKEND    2E34 ENDEND    2E3C TSKT03    2E8A TEST03
  2E92 TB00AA    2EE0 TSTAAA    2EEC TB00BB    2F3A TSTBBB
  2FB6 TSKT04    3004 TEST04    305E $DELAY    306E TSKDLY
  30BC DELAYE    3122 TST05     3170 TEST05    3960 SVCTB
  39CE START     4AFC BUF12     4DDE TSKC1B    4E4C TASKC
  4EF8 OVRTB     4F46 OVRTSK    4FDE OVRLAY    50DE CRDVR
  51C8 CRTRM     527C FLAG04    535C LPDVR     540E LPTRM
  5420 FLAG62    545E DSCDVR    5626 DSCTRM    5720 FLAGC6
  5820 FILE20    5826 FILE21    582C FILE22    8005 INTRVL
  FFFB MSTIM

COMMON-BLOCKS:
NONE

UNDEFINED:
NONE
```

Figure 4. Memory Map (As Listed by the OS Library Loader)

LABEL LU xxxxxx

This command is used in conjunction with adding programs to a library file. If the program to be added to the library does not have a label, then this command can be used to give a program a label in the file. The command causes the loader to output one record to logical unit LU containing the program label xxxxxx. The label must not exceed 6 alpha-numeric digits. If it does, only the first six are used. This command could be used prior to an assembly or compilation which writes the binary object program to logical unit LU.

OUT LU xxxxxx

This command selects the output mode which is used to generate a load module rather than load programs into memory. In this mode, during LOAD, LINK, and EDIT operations, all data that would normally be loaded into memory is output in loader format to logical unit LU. All programs are output as absolute code, origined at the value of Bias in effect when the load module is created. Any external references or common references are resolved and converted to forward references without symbolic names. Thus, a load module program can be loaded by any INTERDATA Loader, including the REL Loader or BOSS or DOS Resident Loader. By resetting the value of Bias for the load module, below the top of the OS Library Loader, and loading the load module with the Resident Loader, a user can effectively overlay the OS Loader. Loading the load module with the Resident Loader, a user can overlay the OS Library Loader at run time.

Prior to selecting the output mode, if a program Label is specified in the command, a label record with the name xxxxxx is generated as described in the LABEL command.

XOUT

This command, which has no argument, generates the final record of the load module, and cancels the output mode. The command XOUT is used in conjunction with generating a load module. A typical command sequence to create a load module would be OUT, BIAS, LOAD, LINK, EDIT, and XOUT.

REWIND LU

This command rewinds logical unit LU.

BR LU

This causes the file on LU to be repositioned backward one logical record.

BF LU

This moves the file on LU backwards to a file mark.

FR LU

This command moves the file on LU forward one logical record.

FF LU

This command moves the file on LU forward to a file mark. The file is left positioned immediately past the file mark.

COPY LULU xxxxxx

This command causes one program to be copied from the first logical unit to the second. If a program Label is specified in the command, the first logical unit is searched for program xxxxxx as described under FIND. If no Label is specified, the first program encountered on the first logical unit is copied. Following a COPY operation, the first logical unit specified is left positioned past the end of the program copied.

WF (or EOF)

This command writes an end of file to logical unit LU and then backspaces over the file mark.

DUPE LULU xxxxxx

This command causes programs to be copied continuously from the first logical unit to the second until program xxxxxx is found. The specified program is not copied, and the operation terminates. If no Label is specified, programs are copied until an end of file or device end on the first logical unit is detected.

To duplicate an entire file of programs from logical unit 1 to logical unit 2 the following command sequence is sufficient.

```
REWIND    2
REWIND    1
DUPE    0102
```

TABLE LULU

This command scans the first logical unit and lists all program labels on the second logical unit. Thus a table of contents of a file containing many labeled programs may be obtained. Any program in the file without a label is not shown in the list.

END

This command returns control to the operating system. It may be issued at any time and the Loader may subsequently be restarted from its origin without losing any previously loaded data. The Loader tables are initialized only when executing a LOAD Command. It is therefore possible to exit the loader temporarily (such as to reassign an I/O device) and return to it during a sequence of related operations.

OV (DOS OVERLAY)

The form of this command is OV XXXX where XXXX = the sum of all active storage areas requested in the program via SVC 2 (Get Storage). If not specified, a default value of X'168' is assumed. This command is used to indicate that a subroutine about to be linked to a program is used as an overlay subroutine under DOS. The OV command, used prior to linking a subroutine and subsequent subroutines, causes them to occupy a common area of memory (i.e., each subroutine is originated at the first available location above the main program). The OV can only be used with the OUT feature of the Library Loader to create overlay modules. These modules may be called by a FORTRAN program through subroutine IFETCH and by assembly language programs through an SVC 5.

The OV command's operand specifies the area to be reserved between the main program and the first overlay. The operand is given a number of bytes (hex) rounded down to a halfword value. The default value (X'168' bytes) is the amount of storage requested by an executing FORTRAN program; i.e., the operand need not be specified for FORTRAN overlay operations. For assembly language programs, the operand should specify the sum of the active storage areas requested. An operand of zero in this command will delete the reserved area. (EX: OV 0).

Ex: If in a program, the following storage requests were made:

```
GET X'100'
GET X'200'
RELEASE X'200'
RELEASE X'100'
```

then the OV argument is X'100' + X'200' = X'300'.

If requests are:

```
GET X'100'
RELEASE X'100'
GET X'50'
RELEASE X'50'
```

then the OV argument is X'100'.

As an example of overlay establishing procedures, let us assume that we have a FORTRAN application program and 2 subroutines which are to be linked as overlays. The program and its subroutines have been compiled and their object programs are on paper tape. Further, for purposes of this example, the main program calls the first overlay via Logical Unit 2 and the second via Logical Unit 3.

In the following example, Logical Unit 4 is assigned to the Paper Tape Reader, Logical Unit 6 is assigned to the FORTRAN Run-Time Library device, and Logical Units 1, 2 and 3 are assigned to the output devices which will contain the main program, Overlay 1, and Overlay 2 modules respectively.

Use relevant positioning commands (FF, BR, etc.) to position output files on LUs 1, 2, and 3.

OUT	1	Set the OUT option for Logical Unit 1.
LO	4	Load the main programs.
ED	6	Edit with the Run Time Library.

```
XOUT
WF      1       Write file mark on output file.
OUT     2       Set the OUT option for Logical Unit 2.
OV              Indicate an overlay is to be linked.
LI      4       Link First Overlay.
ED      6       Edit with the Run Time Library.
XOUT
WF      2       Write EOF.
OV              Indicate another overlay is to be linked.
OUT     3       Set the OUT option for Logical Unit 3.
LI      4       Link Second overlay.
ED      6       Edit with the Run Time Library.
XOUT            Terminate OUT option.
WF      3       Write EOF.
EN              Terminate Library Loader.
```

The first OV command following a LOAD command establishes the BIAS location for all overlays to follow.

LG (LOG)

This command has the form

LG XUU

where UU is the logical unit of a list device. All loader commands read from the loader command input device (Logical Unit 5) are logged on Logical Unit UU. (If UU=0, no logging occurs.) X may be a digit of 0 or 1 and is used to suppress the message 'LOADER' after each command in cases where commands are not being read from the Teletype.

Examples:

```
LG      4       Log on Unit 4.
LG      103     Log on Unit 3, suppress 'LOADER' message.
LG      3       Continue logging, reinstate 'LOADER' message.
LG      0       Stop logging, reinstate 'LOADER' message.
```

PAUSE

This command causes the loader to issue an SVC 2 requesting the PAUSE option. The loader may subsequently be restarted by the operator command CONTINUE. Upon loader continuation, the status of the loader is the same as prior to the pause.

TOCOR bbbb

This command is only valid immediately after an OUT command. It sets the top-of-core address (for loading of FORTRAN COMMON) to bbbb.

The following commands are additionally available in the F01 version only:

DIRECT LUSZ

This command is used to initialize an OS/16-MT Overlay File on Logical Unit LU. SZ sectors are reserved for the overlay directory (F01 only).

OLAY LU xxxxxx

This command is used to prepare for loading an OS/16-MT overlay to be written to the Overlay File on LU. xxxxxx must have been previously defined as an ENTRY (F01 only).

TTASK LU xxxxxx

This command is used to prepare for loading an OS/16-MT Transient Task to be written to the Transient Task File on LU. Transient Tasks and overlays may be intermixed on the file so long as unique names are adhered to. The OS/16-MT Main Program must have previously been loaded, and the Transient Task area, TAREA, defined as an entry, if operating under BOSS; if operating under OS/16-MT, all symbols in the Reentrant Library Symbol Table (RLSTAB) are defined as ENTRYs and may be referenced by Transient Tasks or overlays (F01 only).

INSERT

This command causes all programs LInked (or EDited) since the last OLAY or TTASK command to be inserted in the Overlay/Transient Task File specified in the OLAY or TTASK command. (F01 only).

CMPRESS LU

This command causes the directory on LU to be compressed, to optimally use space lost by previous DELETE commands (F01 only).

MD LULU

This command causes a map of the overlay file directory on the first LU to be output to the second LU. If only one LU is specified, the directory is obtained from the overlay file accesses by the last DI, OL, DE, or CM command (F01 only).

4.2 Library Loader Messages

MEM-FULL

This message is logged if a program being loaded exceeds the available memory. This may happen if too little Common space is allocated (if the program defined Common) or if too large a program is loaded. A table of Labels, Common, and external symbol names is kept, building downward from the bottom of Common (See Memory Maps). If the program overlaps the table, the above message is logged, and the load is aborted.

CMD-ERR

This message is logged when the loader does not recognize an operator command, if a Label command is found between an OUT and XOUT commands or if GO is issued when no transfer address has been specified in any loaded program.

* * NO PROGRAMS * *

This message is logged if a MAP is requested and no labeled programs have been loaded.

EOF

A file mark or END VOL has been read during binary input.

EOM

An END OF MEDIA has been encountered during a binary input.

CKSM ERR

A record has been read in which the checksum does not match the checksum computed by the loader. Following this message, the loader pauses. If the input device is Paper Tape, it may be manually backspaced one record before continuing the load. Magnetic Tape is automatically backspaced on continue.

SEQ ERR

A record has been read out of sequence. Reposition the input unit and continue.

REF-LOOP xxxxxx

A loop has been found in a forward or external reference thread, if external, involving xxxxxx. The input tape has been generated incorrectly.

M xxxxxx

An entry point (xxxxxx) has been defined more than once. The load continues, and the previous definition remains in effect. The new definition is ignored.

LOAD ABORTED

An unrecoverable error condition has caused a load in progress to be aborted. It is necessary to repeat the original LOAD command, as the table may contain incorrect symbol locations.

ADRS-ERR

A program has attempted to load below the top of the Loader. The program may contain absolute data at such an address, or the BIAS may be set incorrectly. The load process is aborted.

DEV END

A device went off line during input.

I/O DEV ERR

A parity error has been detected on input or an off line condition has been detected on output. If it is an input error, the OS status byte and device address in hex-notation appear on the next line.

LOAD-ERR

An illegal control item was detected during a load.

xxxxxx NOT FOUND

An end of device condition has occurred before locating the label xxxxxx during a search operation.

The following messages are additionally available in the OS Library Loader F01 version only:

OVLY NAME ERR

A name was specified in the OLAY command that had not been defined in a previously loaded program, or an attempt was made to DELETE a program not in the directory.

UNDEF EXTRN IN OV

An attempt was made to INSERT an overlay before all EXTRN's in the overlay had been satisfied.

DIR FORMAT ERROR

A program name containing an invalid ASCII character was encountered during an MD operation.

DIR FULL

The number of programs (including those deleted since the file was last CMPRESSed or initialized) in the overlay file is equal to 25 times the number of sectors reserved for the directory.

IN W/O OL

An INSERT command was issued without a corresponding OLAY command.

4.3 OS/16-MT Overlay and Transient Task Procedures

The OS Library Loader may be used to create, maintain, and manipulate a file of binary, core-image overlays (which may be loaded by the OS/16-MT SVC 5 supervisor call) and Transient Tasks (which may be loaded by the LOAD operator command). The file must be resident on a random-access device, either disc or drum (if a drum is used, its pseudo-sector size must be set to 256). The DIRECT command is used to initialize a new File, and establish the number of sectors to be used for the Overlay/Transient Task Directory.

Overlays may only be added to the file if the overlay name, and all EXTRN's and ENTRY's contained within the overlay, are present in the Loader's Symbol Table. This means that the root segment of the overlay must have either been loaded into memory, or output in a load module by the OS loader. The overlay must have been defined as an ENTRY and storage reserved (in multiples of 256-byte sectors) within the root segment.

Transient Tasks differ only in that the symbol used to define the load address (and which therefore must have previously been defined as an ENTRY) is fixed at "TAREA", the start of the Transient Task area within the OS/16-MT main Program. This is accomplished automatically, via the Reentrant Library Symbol Table, when operating under OS/16-MT itself; it can also be done by building an OS/16-MT load module under BOSS.

After the overlay or Transient Task has been inserted in the file, ENTRY's encountered within it are removed from the OS Loader's symbol table; hence, no program loaded subsequently may reference the overlay or Transient Table. The proper sequence for inserting an overlay or Transient Task is first to issue an OLAY or TTASK command to set up for the load; second, LINK or EDIT all required modules; and third, issue an INSERT command to add the overlay or Transient Task to the file.

The program is actually loaded into memory as part of this process--if the root segment was being processed in the output mode, immediately above the OS Loader, destroying any programs in this area. The amount of memory overwritten is equal to the size of the largest program loaded. Otherwise, it is loaded at the address specified in the root segment. The CMPRESS function also uses the area immediately above the OS Loader, overwriting an area equal to the largest program in the file, rounded up to the next highest multiple of 256 bytes, plus an additional 256 bytes. Issuing either a DELETE, CMPRESS, MD, or another OLAY or TTASK command while processing an overlay or Transient Task results in a CMD ERR printout.

As an example, take two FORTRAN application programs, one to be established as a Transient Task, TTASK1, and another with two subroutines which are to be linked as overlays, OVER1 and OVER2. The programs and subroutines have been compiled and their object modules are on paper tape. The Run Time Library is resident on a disc file which has been assigned to Logical Unit 7. The OS/16-MT main program, SYSTAB, RLSTAB, and the resident task's Task Block/Get Storage module have been assembled and are resident on a disc file which has been assigned to Logical Unit 6. Logical Unit 1 is assigned to the Paper Tape Reader, and Logical Unit 2 is assigned to the load module output device. The Overlay/Transient Task file has been assigned to Logical Unit 4.

```
OUT 2           Set Output Mode LU 2
LO 6 OS16MT     Load OS/16-MT main program
LI 6 RLSTAB     Link RLSTAB
LI 6 SYSTAB     Link SYSTAB
LI 6 TSKTB1     Link TB/Get Storage module
LI 1            Link Resident Task
ED 7            Edit Run Time Library
TT 4 TTASK1     Set up for Transient Task
LI 1            Link Transient Task
ED 7            Edit RTL
IN              Insert Transient Task
OL 4 OVER1      Set up for 1st overlay
LI 1            Link overlay
ED 7            Edit RTL
IN              Insert Transient Task
OL 4 OVER2      Set up for 2nd overlay
LI 1            Link overlay
ED 7            Edit RTL
IN              Insert overlay
XOUT            Terminate Load Module
EN              Terminate Library Loader
```

4.4 Functional Variations

Two functional variations of the OS Library Loader are now being made available. The first, F00, is functionally similar to previous versions of the Loader, occupies approximately 4.5K bytes of memory, and uses the Model 4 instruction set. The second, F01, contains support for OS/16-MT overlay operations, occupies approximately 7K bytes of memory, and uses the Model 70 instruction set. F00 should be used whenever memory is critical. F01 must be used to support OS/16-MT overlay. They comprise two out of the four possible variations of the Loader which may be generated via the two System Generation parameters which have been introduced. They are:

```
DOS       EQU 0    deletes DOS overlay support
DOS       EQU 1    includes DOS overlay support

OS16MT    EQU 0    deletes OS/16-MT overlay support
OS16MT    EQU 1    includes OS/16-MT overlay support
```

F00 is obtained by setting:

```
DOS       EQU 0
OS16MT    EQU 0
```

F01 is obtained by setting:

```
DOS       EQU 1
OS16MT    EQU 1
```

These SYSGEN equates are specified at assembly time. The CAL Assembler will attempt to read the equates from Logical Unit 7 via the CAL copy feature. The equate statements are normally entered through the Teletype, and must be followed by a "/*" which terminates the copy and continues the assembly.

5. OS RANDOM ACCESS BOOTSTRAP 07-046

The OS RANDOM ACCESS BOOTSTRAP PROGRAM loads the memory image of the BASIC OPERATING SYSTEM (BOSS), 03-019, previously written to disc or floppy disc and branches to location X'2D0'. The program is loaded by the 50-sequence with locations X'78' through X'7E' set up as follows:

LOCATION	CONTENTS	MEANING
X'78'	XXXX	BINDV (Standard 50-sequence format)
X'7A'	DDZZ	DD = DISC or FLOPPY DISC DRIVE Address ZZ = Code Valid Codes: X'00' = 2.5 or 10MB Disc X'37' = Floppy Disc X'04' = 40MB Disc
X'7C'	CCSS	CC = Controller address for disc = 0 for Floppy Disc SS = Selch Address for disc = Floppy disc drive number (0-3)
X'7E'	SSSS	Sector Address on disc.

When the bootstrap is loaded, it performs a top-of-memory search and relocates itself to the top X'196' bytes of memory. The program then reads from the specified device and sector the memory-image binary data of BOSS into memory from location 0 to the top-of-memory minus X'196' bytes. If any errors are detected during the load process, the program halts with the PSW set for restarting at the initialize routine.

APPENDIX 1

SUMMARY OF OS LIBRARY LOADER COMMANDS

CMD:	ARG1:	ARG2:	MEANING:
AMAP	LU		Output Memory-Map to LU in alphabetical symbol order
BC	LLLL		Set Length of Blank Common
BF	LU		Backspace LU to file mark
BIAS	BBBB		Set Bias to BBBB
BIAS			Set Bias to End of Loader
BR	LU		Backspace LU one record
COPY	LU LU		Copy One Program from LU A to LU B
COPY	LU LU	NAME	Find and Copy Named Program
DUPE	LU LU		Duplicate Until EOF or Dev END
DUPE	LU LU	NAME	Duplicate Until Name is Read
EDIT	LU		Load only Referenced Programs from LU
EDIT	LU	NAME	Find Named Label and Edit
END			Exit to the OS
EOF	LU		Write a File-Mark on LU
FF	LU		Skip to file mark on LU
FIND	LU	NAME	Position LU to Start of Named Prog.
FR	LU		Skip 1 record on LU
GO			Transfer to Loaded Program
LABEL	LU	NAME	Write Label-Record on File LU
LC	LLLL		Set Length of Labeled Common
LF	LU		Link in entire file
LF	LU	NAME	Link in entire file from NAME to end
LG	XLU		Set command log option to LU

CMD:	ARG1:	ARG2:	MEANING:
LINK	LU		Load One Program from LU, Linking
LOAD	LU		Purge Table, Load One Prog. from LU
LOAD	LU	NAME	Find and Load a Program
MAP	LU		Output Memory-Map to LU in memory address order
OUT	LU	LABEL	Output Labeled Load-Module to LU
OUT	LU		Output Load-Module to LU
OVLY	XXXX		Set Get Storage Space to XXXX
PAUSE			Pause Loader
REWIND	LU		Rewind LU
RW	LU		Same as RE
TABLE	LULU		Write Table-of-Contents to LU
TOCOR	BBBB		After an OUT, sets top-of-core address for loading of FORTRAN Common to BBBB
WF	LU		Same as EOF
XOUT			End Load-Module, Write Last Record

The following commands are additionally available in the F01 version only:

DIRECT	LUSZ		Initialize Overlay File, reserving SZ Sectors for Overlay Directory
OLAY	LU	XXXXXX	Prepare to load overlay XXXXXX to be written to LU
TTASK	LU	XXXXXX	Prepare to load Transient Task XXXXXX to be written to LU
INSERT			Insert all programs since OLAY, TTASK command in overlay file on LU
CMPRESS	LU		Compress directory on LU
MD	LULU		Output map of directory on LU A to LU B
MD	LU		Output map of directory on last referenced overlay file to LU

APPENDIX 2

SUMMARY OF STAND ALONE LOADER OPERATION

All values below are expressed in hexadecimal:

Starting address after boot load*	nC00	nA00
Restart Address in general#	ORG	ORG
Bias define address	ORG+8	ORG+8
Bias definition value	ORG+A	ORG+A
Continue address	ORG+26	ORG+26
Approximate Loader size	X'400'	X'600'
Illegal control items	C, E0-E3, E5-E7	E0-E3, E5-E7
Ignored control items	D, F	---------

*n = 0, 1, 2, 3...for memory sizes 4K, 8K, 12K, 16K, etc.

#General Loader Restart - sets BIAS to X'80'
- clears symbol table
- clears any transfer address

The following Display indications are appropriate to the REL Loader.

Display Lights	Condition	Comment
XX00	Normal End	Load Complete
XX01	Checksum Error	Following checksum or sequence number error reposition Paper Tape and depress EXECUTE to re-read. If Magnetic Tape or Cassette - parity error will cause 2 retries before returning this status.
XX02	Sequence Error	
XX03	Attempt to load over loader	
XX04	Ref-Chain Loaded	
XXFn	Load Error	Improper control item detected where n is the bad item. Depress EXECUTE to ignore the data and continue. Refer to Section 2 for a definition of loader control, control items.

APPENDIX 3

50 SEQUENCES

MODEL 3 50 SEQUENCE

X'0050'	C820	LOAD	LHI	2,X'80'	8-BIT LOADER
52	0080				
54	C830		LHI	3,1	
56	0001				
58	C840		LHI	4,X'CF'	
5A	00CF				
5C	D3A0		LB	10,BINDV	
5E	0078				
60	DEA0		OC	10,BINDV+1	
62	0079				
64	9DAE	SENSE	SSR	10,14	
66	08EE		LHR	14,14	
68	4230		BTC	3,SENSE	
6A	0064				
6C	DBA2		RD	10,0 (2)	
6E	0000				
70	C120		BXLE	2,SENSE	
72	0064				
74	4300		B	X'80'	
76	0080				
78	0294	BINDV	DC	X'0294'	DEVICE TABLE
7A	0298	BOUTDV	DC	X'0298'	
7C	0294	SINDV	DC	X'0294'	
7E	0298	LISTDV	DC	X'0298'	

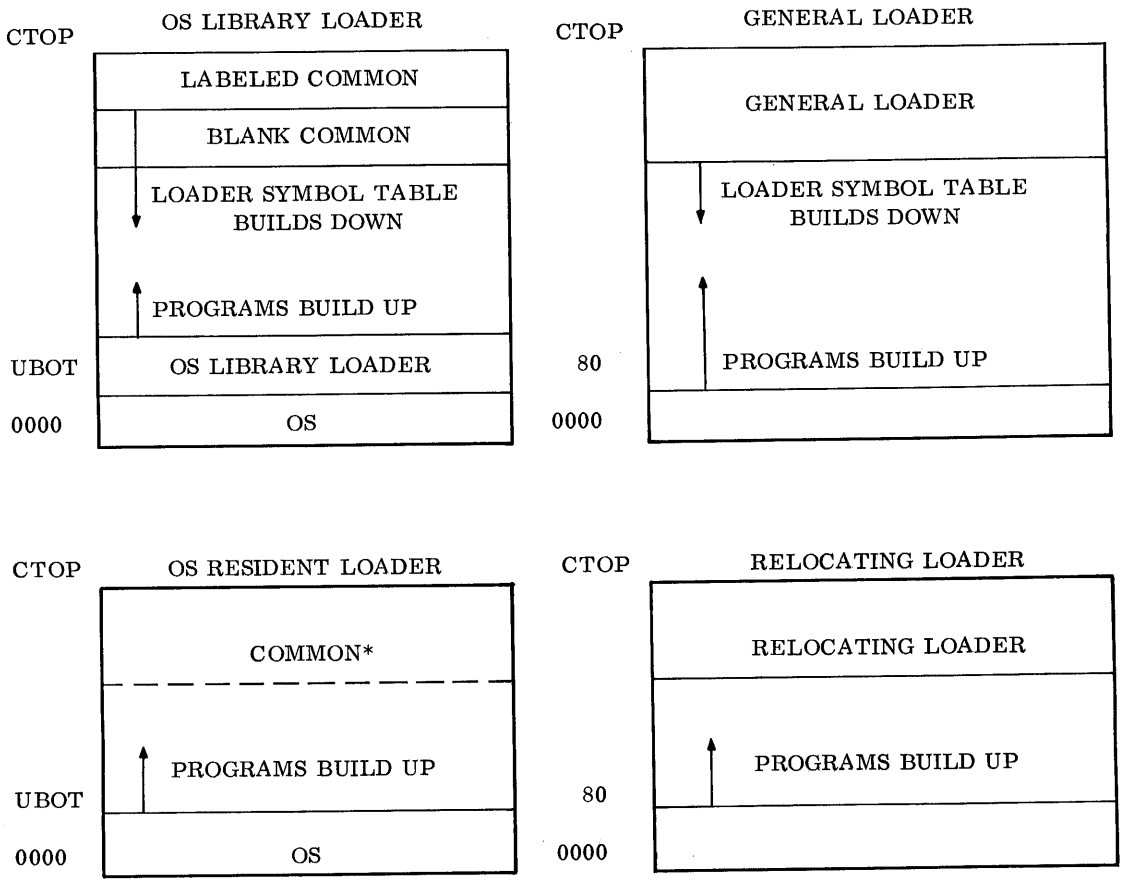
MODEL 4, 5, 50, 70, 74, 80, 85 50 SEQUENCE

```
X'0050'   D500            AL      X'00CF'
    52    00CF
    54    4300            B       X'80'
    56    0080
    78    0294  BINDV     DC      X'0294'     DEVICE TABLE
    7A    0298  BOUTDV    DC      X'0298'
    7C    0294  SINDV     DC      X'0294'
    7E    0298  LISTDV    DC      X'0298'
```

	TTY	HSPTR/P	CAS	MT	Floppy Disc
BINDV '78'	0294	1399	45A1	85A1	C186

APPENDIX 4

LOADER MEMORY MAPS



CTOP = TOP OF CORE (LAST ADDRESS WITHIN MEMORY).

* COMMON USED ONLY IF LOADING A LOAD MODULE PRODUCED BY OS LIBRARY LOADER.

APPENDIX 5

ASSEMBLY/CONSTRUCTION PROCEDURES
FOR
REL LOADER, 06-024R03
AND
GENERAL LOADER, 06-025R03

1. Load the Relocating or General Loader using the "50 Sequence".

2. Load the Bootstrap/Fast-Format Loader (06-030R02M16) using a loader bias of X'1000' with the loader selected in Step 1.

3. Start the program loaded in Step 2. It outputs the Bootstrap/Fast-Format loader to the device whose address and command are specified at X'7A'.

4. Load the Rel Boot Loader (03-031R01M16) using the loader selected in Step 1. The Rel Boot Loader is an absolute program and loads at X'80'.

5. Change the halfword at location X'A6' to X'400' for the Rel Loader Bootstrap or X'600' for the General Loader Bootstrap.

6. Load the Fast Format Puncher (06-031R02M16) at X'1000' using the loader used in Step 1.

7. Change the three consecutive halfwords at X'1000' to X'80' (start address), X'2D0' (end address) and X'80' (transfer address). Start the Fast-Format Puncher at X'1010' and it will output absolute fast-format records of the Rel Boot Loader to the device whose address is specified at X'7A'.

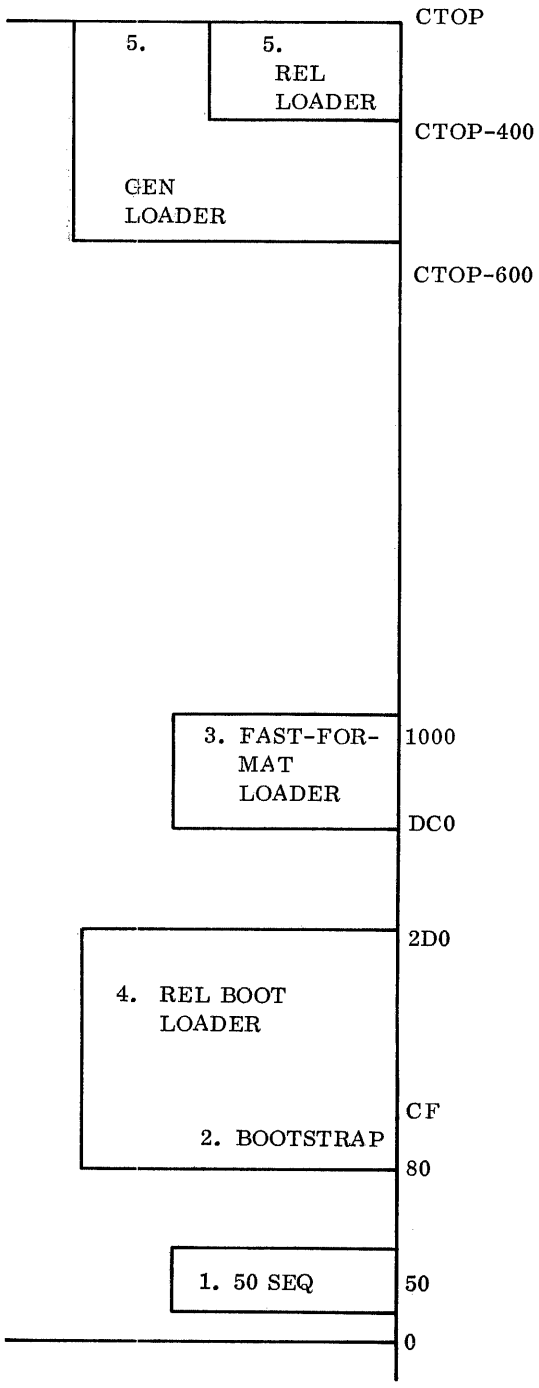
8. Copy the loader to be bootstrapped (M16 format) to the output device by loading the operating system and using the functions of DOS COPY or OS COPY. Alternatively, the assembler may be loaded, and the loader assembled. The object program output completes the bootstrap tape.

BOOTSTRAPPING PROCEDURES FOR REL LOADER, 06-024R05

The Relocating Loader, R05, can be bootstrapped to any device, (Paper tape, Mag tape or cassette), by using the OS/16 Boot Puncher program, 03-108.

EXECUTION MAP

AT LOAD TIME OF THE BOOT-STRAPPED VERSIONS
OF THE REL AND GENERAL LOADERS



APPENDIX 6

BULK STORAGE FILE PROCEDURES FOR R05 AND LOWER

The OS Library Loader may be used to create, maintain, and manipulate a file of programs, using the set of operator commands described. The individual programs in the file must be in standard loader format, and each program must be labeled. Following the last program, there must be a file mark. Each of the commands which may be used to write out a binary object record is designed with Magnetic Tape in mind. Before any record is written by the LABEL, COPY, DUPE, and OUT directives, the output file is positioned to the gap immediately preceding the file mark. This is done by searching forward for the mark, and then backspacing over it. After writing the final record of a program or load module, a file mark is written, and the unit backspaced one record. It should be pointed out that these functions have no effect on non-bulk storage devices; they are ignored. Because these directives never allow a write to a library file without first positioning it past its end of information, they may only be used to add to an existing library.

NOTE

The RTOS/OS 16-MT Drum Driver does not support a write file mark command and therefore the Magnetic Tape oriented commands will not function properly.

To create a new file of programs, the REWIND and EOF commands are used to place an initial file mark on the beginning of the file. Once this is done, programs may be copied onto the file by using the COPY or DUPE Commands. If the program to be copied onto the file does not already have a label, a preceding label may be written before it using the LABEL Command. As programs are added to the file, the mark is propagated along, following the last one.

Any time the library is read, and a file mark is encountered, the operation is terminated immediately, the message EOF is logged, and the tape is backspaced over the mark. If a LOAD, LINK, EDIT, FIND, COPY, or DUPE terminates with EOF, it indicates that the system was searching for a Label not found on the tape being searched. It is the normal termination for the TABLE Command.

If it is desired to add a program to a file with some other program (such as the output from the assembler) the file should first be positioned. Then the new material should be added and the EOF Command should be issued. If the LABEL Command is issued prior to the assembly, it leaves the file positioned past the Label written.

Library Creation

To create a file of programs from Paper Tape to a bulk storage device the following procedure should be followed. Assume the Paper Tape Reader has been assigned to Logical Unit 1 and a Magnetic Tape unit has been assigned to Logical Unit 2.

a) Load the OS Library Loader Program with an operating system and execute.

b) Rewind or otherwise position the output file.

c) Copy the OS Library Loader Paper Tape to the Magnetic Tape.

```
CO 102
```

d) Copy any other desired programs such as the OS ASSEMBLER, the FORTRAN compiler, or any user written program.

```
CO 102
```

e) Copy the RUN TIME LIBRARY Paper Tape to Magnetic Tape with the command.

```
DU 102 ENDVOL
```

This command should be repeated once for each separate input Paper Tape.

f) Other programs may be added on to the end of the program file at a future time with the CO or DU commands.

NOTE: User written subroutines may be edited from the library if the following conditions are met:

1) The subroutine contains a LAB = field in its OPTION card, or a label is created with the LABEL command.

2) The subroutine contains an ENTRY statement with a name that is the same as the LABEL option.

3) The user written subroutine contains an EXTRN statement which refers to a name which corresponds to the label in the library.

APPENDIX 7

REVISION INFORMATION

Level R06 of the OS LIBRARY LOADER is a major departure from the R05 level program. Principally, the implicit positioning contained in the Magnetic Tape oriented commands, (LABEL, COPY, DUPE, etc.) has been deleted in R06. All versions of the Loader R06 and above log their revision level when entered. For those users with R05 and lower, this document may still be used if the section on bulk storage procedures (now moved to Appendix 6) is consulted.

The LINKFILE Command has been implemented in R06. This command is an internal iteration of the LINK command. It executes LINKs until an EOF, EOM on program ENDVOL is encountered. Its intention is to simplify the loading of a Main program and its subroutines when present on a single file of object programs.

The commands WF (Write File Mark) and RW (Rewind) have been added for compatibility with DOS command syntax. The old commands of EOF and RE are still available.

INDEX

PUBLICATION COMMENT FORM

Please use this postage-paid form to make any comments, suggestions, criticisms, etc. concerning this publication.

From _____ Date _____

Title _____ Publication Title _____

Company _____ Publication Number _____

Address _____

_____

_____

FOLD FOLD

Check the appropriate item.

☐ Error Page No. _____ Drawing No. _____

☐ Addition Page No. _____ Drawing No. _____

☐ Other Page No. _____ Drawing No. _____

Explanation:

FOLD FOLD

CUT ALONG LINE

Fold and Staple

No postage necessary if mailed in U.S.A.

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT No. 22
OCEANPORT, N. J.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY:

INTERDATA®

Subsidiary of PERKIN-ELMER
Oceanport, New Jersey 07757, U.S.A.

TECH PUBLICATIONS DEPT. MS 322

FOLD FOLD

STAPLE STAPLE