10.

PREFACE

The DGC relocatable loader, for the ECLIPSE® line computers, loads and links any number of relocatable binary or absolute binary programs produced by the DGC macro assembler, relocatable assembler, or absolute assembler, plus selected libraries of relocatable programs.

In relocatable assembly, storage words are assigned a relative location counter value. The value is initially zero and is incremented for every storage word generated. At the termination of assembly, if n words are generated, they are assigned to relative locations 0, 1, ..., n-1. The actual addresses assigned to words generated are determined by the relocatable loader.

The loader maintains the value of the first location available for loading, based on the programs previously loaded. As each assembled program is loaded, the relocatable loader updates the value of the first location available for loading. In this way, any number of separately assembled modules can be loaded together without any conflict in absolute storage assignment.

Library files are simply collections of relocatable binary programs, one or more of which will be loaded to resolve external references appearing in previously loaded programs.

There are three versions of the relocatable loader. The stand-alone relocatable loader is supplied in absolute binary, tape 091-000038, and is used for stand-alone systems using paper tape I/O. The SOS magnetic tape/cassette relocatable loader is supplied in relocatable binary form, tape 089-000120, and is used for loading under SOS systems that use either magnetic tape or cassette I/O. The relocatable loader used under the Real Time Disk Operating System, RDOS, is supplied as a dump tape, 088-000049.

When the relocatable loader is invoked, relocatable binary programs, absolute binary programs, and libraries will be loaded in the order in which they are given to the loader. In the case of the stand-alone relocatable loader, this will be the order in which they appear on the input device. In the case of other versions of the loader, the order of loading is the order in which the names of the programs appear in an RLDR command line. Each library is searched once each time its name appears in the command line.

TABLE OF CONTENTS

CHAPTER 2 - SOS MAGNETIC TAPE/CASSETTE RELOCATABLE LOADER (Continued)

CHAPTER 3 - RDOS RELOCATABLE LOADER

APPENDIX A - RDOS OVERLAY LOADER

APPENDIX B - RLDR SYMBOL TABLE FORMATS

CHAPTER 1

STAND-ALONE RELOCATABLE LOADER

OPERATION

The relocatable loader tape is in binary format, and is loaded using the binary loader (091-000004). Once loaded, the relocatable loader will self-start and type:

SAFE=

on the Teletype. This queries the user about the octal number of words to be preserved at the high end of memory.

The default response is a carriage return, which will cause the loader to save 200 octal words - enough to preserve both bootstrap and binary loaders.

Otherwise, the user may input an octal number, terminated by a carriage return, giving the number of octal words to be saved. The user response may be up to 5 octal digits long and must be within the limits of memory.

An error on input will cause the loader to repeat the query, SAFE=; the error cases are:

1. A character other than an octal digit or a carriage return is input.

2. More than five octal digits are input.

3. The number specified is too large for the user's core configuration, that is, no load space remains.

When the SAFE= query has been correctly answered, the value specified is fixed for the duration of the loading process. The loader will then prompt a user response by typing:

*

on the teletypewriter.

Relocatable loader action is initiated by single digit responses to the loader prompt, *. The possible loader mode responses are tabulated on the following page and are described throughout this manual. Each time the loader has completed its response to a user request, it will type

*

OPERATION (Continued)

and await a new request. If an illegal response is input, the loader prints:

?

and awaits a legal response. User-loader interaction is terminated by responding to a prompt with the digit 8 (terminate loading process and prepare for execution).

To reinitialize the loading process if the process was terminated by a fatal load error, the user may issue the digit 7 in response to the * query.

TABLE OF RESPONSES TO LOADER PROMPT *

Response	Effect
1	Load a relocatable binary or a library tape from the teletype reader.
2	Load a relocatable binary or a library tape from the paper tape reader.
3	Force a loading address for normally relocatable code.
4	Complement the load-all-symbols switch.
5	Print current memory limits.
6	Print a loader map.
7	Reinitialize the loader.
8	Terminate the load process to prepare for execution.
9	Print all undefined symbols.

RESTART PROCEDURES

To restart the loading process, the user may press RESET, enter 000377 in the data switches, press START.

INPUT TO THE STAND-ALONE LOADER

The input to the relocatable loader is in the form of relocatable binary tapes output from the relocatable assembler. Each tape is divided into a series of blocks and

must contain at least a title block and a start block. The order of blocks input to the loader is shown below. Each block type is described in detail later in this manual.

Title block
COMMON block(s)
Entry block(s)
.CSIZ block(s)
Displacement External block(s)
Relocatable data block(s) Global addition block(s) Global start and end block(s)
Normal external block(s)
Local symbol block(s)
Start block

To load a single relocatable binary tape, the user responds to the "*" prompt with:

1 - input from teletype reader, or

2 - input from paper tape reader

The binary tape will be loaded, and the loader will respond "*" after the start block has been processed. The user can then input another relocatable binary tape or give one of the other responses to the prompt.

RELOCATION VARIABLES ZMAX AND NMAX

The load addresses input to the relocatable loader are in three modes -- absolute, page zero relocatable, and normal relocatable. Absolute origined data is loaded at the locations specified to the assembler. For relocatable origined data, the loader is initialized to assume two relocation variables called NMAX and ZMAX.

RELOCATION VARIABLES ZMAX AND NMAX (Continued)

ZMAX has an initial value of octal 50, where 50 is the first location to be loaded with page zero relocatable data. As each location is filled, ZMAX is updated to reflect the next location available to receive page zero relocatable data.

NMAX has an initial value of octal 440, the first location available to load normal relocatable data. As each location is filled with normal relocatable data, NMAX is updated to represent the next available location.

Determining the Current Values of ZMAX and NMAX

The relocatable loader maintains a symbol table (also called a loader map) which is built down in core from the saved area (response to SAFE=). The current values of ZMAX and NMAX are given in the loader map. The user can obtain the current ZMAX and NMAX values, plus current values of the start and end of the symbol table (SST and EST) and CSZE (unlabeled COMMON size) by responding to the "*" prompt by

5

The first two values given are NMAX and ZMAX in the format:

NMAX nnnnnn
XMAX nnnnnn

where nnnnnn represents the 6-digit current octal value of each variable.

Forcing a Value of NMAX

Before input of a relocatable binary tape, the user can force NMAX to a given value, thus determining the absolute load address for normally relocatable data.

The user can force NMAX to a given value by:

1. Entering the desired octal value in the console data switches, bits 1-15.
2. Responding to the "*" prompt with the digit:

   3

THE SYMBOL TABLE (LOADER MAP)

The symbol table is constructed downward in core from the first address below the

saved area, determined by the SAFE= query.

At the top of the symbol table are entries for NMAX, ZMAX, CSZE, EST, and SST. Below these are all entry symbols, undefined externals, and local symbols (if the load locals switch was set by a response of 4 to the "*" prompt).

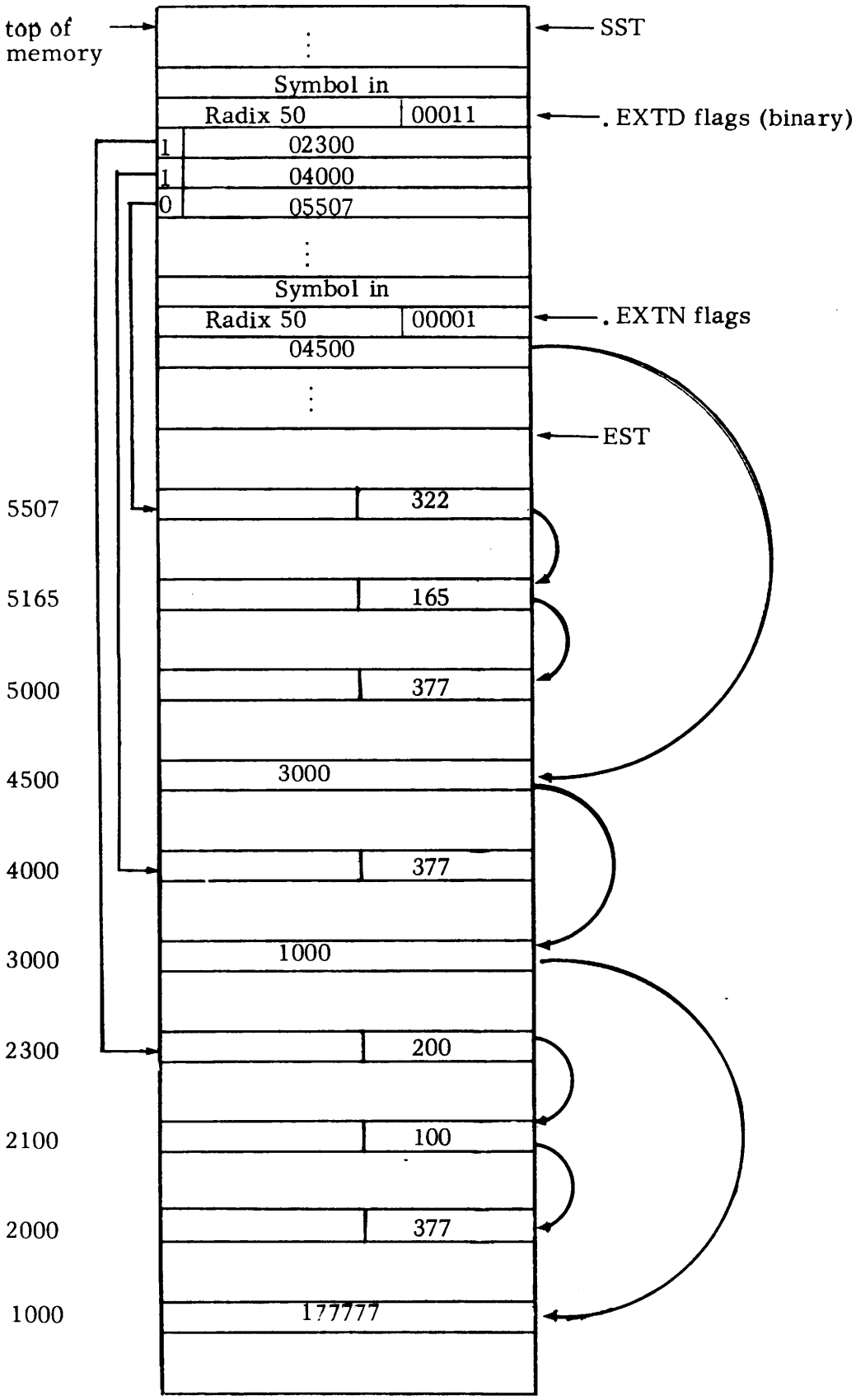
The maximum symbol length is five characters (stored in radix 50 using 27 bits) and there is a six-digit octal value associated with each symbol.

For an entry symbol, its value is an absolute number -- either the core address of the word for which the symbol was a label or the value of the symbol as defined by an equivalence.

For undefined external normals, the number is the absolute address of the last of a chain of references to the symbol. If the number is -1, there were no references. Each reference to the symbol has been replaced by the absolute address of the previous reference, the first reference having been replaced by -1.

For undefined external displacements, there may be more than one reference chain. The value printed is the absolute address at the last reference in the first such chain. The actual symbol table entry has the two-word symbol and the end addresses of n chains, where the first n-1 have bit 0 set and the last does not, signifying the end of the symbol table entry. Within a chain, references are linked via 8-bit relative displacements, contained in the rightmost byte (right half) of each storage word. Each chain is terminated by a word having 377_8 in its right-most byte. Thus, if two consecutive references are farther than 367_8 words apart, a new chain must be started as shown on the following page.

At termination of the load, undefined external normals will be resolved by the relocatable loader to the value -1. Each occurrence of that symbol will be replaced with a -1.


top of memory
SST
Symbol in Radix 50
00011
. EXTD flags (binary)
1 02300
1 04000
0 05507
Symbol in Radix 50
00001
. EXTN flags
04500
EST
5507
322
5165
165
5000
377
4500
3000
4000
377
3000
1000
2300
200
2100
100
2000
377
1000
177777

THE SYMBOL TABLE (LOADER MAP) (Continued)

A symbol may be flagged on the loader map with one of two letters. A U appears on the lefthand side of the symbol if the symbol is an external for which no entry has yet been defined, i.e., an unresolved external.

An M on the lefthand side of the symbol means the symbol is defined in two or more entry or .COMM statements.

An example of part of a symbol table follows. EST means the lowest word of the symbol table - 1. SST means the highest word within the symbol table. CSZE means size of unlabeled Common.

```
TEMP

    NMAX 044723
    ZMAX 006244
    CSZE 000000
     EST 050131
     SST 052001

   AHESZ 000000
   BINAR 006207
   BUFFE 016711
   BUGIN 000000
    CHSW 000174
    CKSQ 000000
   COL01 035622
   COL02 035623
   COL03 035624
   COL04 035625
   COL05 035626
   COL06 035627
   COL07 035630
   COL08 035631
   COL09 035632
   COL10 035633
   COL11 035634
   COL12 035635
   COL13 035636
   COL14 035637
   COL15 035640
   COL16 035641
   COL17 035642
```

THE SYMBOL TABLE (LOADER MAP) (Continued)

To obtain a copy of the symbol table, respond with the digit

6

to the loader "*" prompt.

To obtain a copy of only the undefined symbols in the symbol table, respond with the digit

9

to the loader "*" prompt.

EXECUTION OF LOADED PROGRAMS

Initiation of Execution

Loading of programs is terminated when the user responds

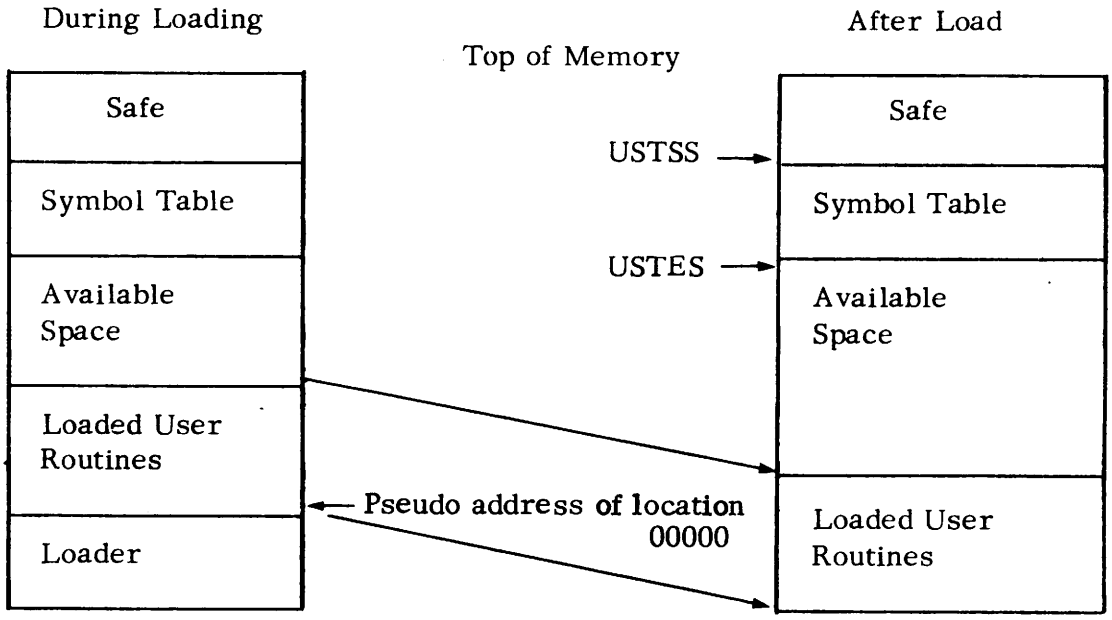
8

to the loader query "*". The programs previously loaded are then moved to reside at the absolute addresses indicated by the loader map. Until the load process is terminated, the loader resides in low core and all programs are loaded assuming a pseudo address for location 00000 which exists above the loader itself. Once loading is terminated, the following occurs:

Location 377 is unconditionally initialized to 2406 (JMP @.+6), providing a convenient restart address. (Location 405 of the User Status Table, UST, is set to the starting address of the loaded core image by the loader. See UST layout on page 1-16.)

Memory is shuffled down to reflect the true addresses as shown on the following page.

EXECUTION OF LOADED PROGRAMS (Continued)



The loader passes information to loaded routines that may be useful for their execution. This information is passed in the User Status Table, which starts at location 400, (See page 1-16).

Starting Address for Execution

After shuffling memory, the relocatable loader will HALT. When the user presses CONTINUE, the loader will HALT again if no starting address has been specified on any of the binary tapes.

If only one of the binary tapes loaded contains a starting address, the address will receive control regardless of the order in which the tapes were loaded.

If more than one binary tape loaded contained a starting address, the last starting address specified by a binary tape will receive control for execution.

LOADING OF LIBRARY TAPES

Library tapes are tapes containing a set of relocatable binaries that are preceded by a library start block and terminated by a library end block. Library tapes are provided by Data General as part of the standard software packages.

Library tapes are loaded in the same way as relocatable binaries. The user mounts

LOADING OF LIBRARY TAPES (Continued)

the library tape in the appropriate input device and responds to the loader "*" query with either 1 or 2.

The library load mode is initiated when the loader encounters a library start block. The loader does not request a new load mode until after encountering the library end block.

The loader will load selected relocatable binary programs from the library tape. Programs in a library tape are loaded only if there is at least one entry symbol defined by that program which corresponds to a currently unresolved external in a previously loaded program. For example, if programs A, B, and C are on a library tape and A calls B which calls C, none of those programs will be loaded unless some program loaded before the library tape has called A. If A has an entry corresponding to a previously unresolved external, then all three programs A, B, and C will be loaded.

LOADING LOCAL AND TITLE SYMBOLS

Local and title symbols are normally loaded only when the user intends to use the symbolic debugger, since the symbols will otherwise occupy symbol table storage space unnecessarily.

The loader maintains a local and title symbols switch which is set by default to inhibit loading of local and title symbols. The user can complement the switch, altering the mode, by responding to the loader "*" query with

4

The loader responds with S when the switch is set to load local and title symbols. The user can complement the switch by issuing another 4, and the loader responds with R, indicating that the switch has been reset.

REINITIALIZATION OF LOADING

If the loading process is terminated by the fatal error (see ERROR DETECTION section), or if the user wishes to start loading over, the loader must be reinitialized. The user can reinitialize loading by responding to the loader query "*" with

7

The loader will then reset ZMAX and NMAX to 50 and 440 respectively, and will reinitialize the symbol table, eliminating all entries.

DETERMINING AVAILABLE CORE

Total core available for program loading is dependent upon loader size, core configuration, the size of the SAFE area, and the number of symbols entered in the symbol table. The following is an approximate formula for determining core available for program loading:

core available $\cong$ sc - 2500 - SAFE - 3*ne

where: sc is the core capability of the system configuration, and

ne is the number of entry points (plus the number of user symbols when in mode 4) defined by all relocatable programs to be loaded.

The quantities are given in octal.

The user can obtain a printout of the current memory limits during loading by giving the response

5

to the loader query "*".

ERROR DETECTION

The loader detects two types of errors -- fatal and nonfatal. Fatal errors prevent further loader action unless the user reinitializes (response of 7). Nonfatal errors do not stop loading but may change the intended state of the user's loaded system.

All errors are indicated by a two-letter code. The code is printed at the teletype followed by a symbol name, if applicable, and by a six-digit octal number, if applicable. The meaning of the octal number is defined later for each of the error codes. The message has the general form:

ee sssss nnnnnn

where ee is the error code.

sssss is the symbol name.

nnnnnn is the octal number.

ERROR DETECTION (Continued)

Nonfatal Errors

Code	Description
DO	Displacement overflow
IB	Illegal block type
IN	Illegal NMAX
ME	Multiply defined symbol
TO	Input timeout
XE	External undefined in external expression

DO Error

```
DO  nnnnnn
```

If, while attempting to resolve an external displacement, the loader finds the displacement is too large, a displacement overflow (DO) error results. The displacement is too large if:

the index = 00 and the unsigned displacement is > 377

or

the index ≠ 00, and the displacement is outside the range:

$$-200 \leq \text{displacement} < +200.$$

The location nnnnnn represents the absolute address where overflow occurred. The displacement is left unresolved with a value of 000.

IB Error

```
IB nnnnnn
```

IB Error (Continued)

If an illegal relocatable block type is read, an illegal block (IB) error results. The octal number nnnnnn represents the block code of the illegal block. The loader will issue a "*" query after issuing the error code. If an improper tape mounted in the reader caused the error, it should be replaced by a relocatable binary or library tape and loading attempted again.

IN Error

```
IN nnnnnn
```

If the user responds 3 to the loader prompt and the value in the switches is lower than the current value of NMAX, an IN error results. The octal number nnnnnn is the illegal value of NMAX. NMAX is unchanged.

ME Error

```
ME sssss nnnnnn
```

When an entry or named common (.COMM) symbol having the same name as one already defined is encountered during loading, a multiply defined entry (ME) error results. The name of the symbol sssss is followed by the absolute address nnnnnn at which it was originally defined.

TO Error

```
TO nnnnnn
```

If the time between input characters becomes excessive, a timeout (TO) error occurs. The usual cause of the timeout error is a binary tape without a start block or a library tape without an end block. The location nnnnnn represents the location in the loader where the timeout occurred. The loader will issue a "*" request when the error occurs.

XE Error

```
XE sssss
```

XE Error (Continued)

If a .GADD block is encountered that references an as yet undefined symbol, an external undefined in external expression (XE) error occurs. Zero is stored in the memory cell. The undefined symbol sssss is printed out following the error indicator.

Fatal Errors

Code	Description
CS	Checksum error
MO	Memory overflow
NA	Negative address
NC	Named COMMON error
OW	Overwrite of memory
XL	External location undefined
ZO	Page zero overflow

CS Error

```
CS nnnnnn
```

If a checksum computed on any block differs from zero, a checksum (CS) error results. The octal number nnnnnn represents the incorrect checksum that was computed.

MO Error

```
MO nnnnnn
```

If the value of NMAX plus the loader size itself conflicts with the bottom of the

MO Error (Continued)

loader's symbol table, a memory overflow (MO) error occurs. The error implies that the user programs are too large to be loaded in the memory configuration. The octal number nnnnnn is the value of NMAX that caused the overflow.

NA Error

NA nnnnnn

If bit 0 of an address word is set to 1, a negative address (NA) error occurs. The assembler restricts addresses to the range:

$$0 \leq \text{address} < 2^{15}$$

A reader error, however, could cause bit 0 to be set. nnnnnn represents the negative address.

NC Error

NC sssss nnnnnn

If two programs have different sizes for a given area of labeled COMMON (defined by .COMM statements), or if the symbol table flags that are associated with the symbol are not 00010_2, a named common (NC) error results. sssss gives the symbol name of the labeled COMMON and nnnnnn indicates the size of the labeled COMMON requested by the present .COMM .

OW Error

OW nnnnnn

The loader does not permit memory cells to be overwritten by subsequent data once they are loaded. If an attempt to overwrite is made, an overwrite (OW) error occurs. The absolute address where the overwrite was attempted is given by nnnnnn.

ERROR DETECTION (Continued)

XL Error

```
XL sssss
```

If a .GLOC block is encountered with data to be loaded at the address of a symbol that is as yet undefined, an external location undefined (XL) error results. The undefined symbol is given by sssss..

ZO Error

```
ZO nnnnnn
```

If in loading page zero relocatable code the code overflows the page zero boundary of 377, a page zero overflow (ZO) occurs. The absolute address of the first word of the data block that caused the overflow is given by nnnnnn .

USER STATUS TABLE OF LOADING INFORMATION

The relocatable loader provides information concerning the loading process in a table called the User Status Table (UST). The UST is origined at location 400: a template is shown below with explanatory information.

```
UST    = 400    ;START OF USER STATUS AREA
USTPC  =   0    ;PROGRAM COUNTER
USTZM  =   1    ;ZMAX
USTSS  =   2    ;START OF SYMBOL TABLE
USTES  =   3    ;END OF SYMBOL TABLE
USTNM  =   4    ;NMAX
USTSA  =   5    ;STARTING ADDRESS
USTDA  =   6    ;DEBUGGER ADDRESS
USTHU  =   7    ;HIGHEST ADDRESS USED BY LOAD MODULE
USTCS  =  10    ;COMMON AREA SIZE
USTIT  =  11    ;INTERRUPT ADDRESS
USTBR  =  12    ;BREAK ADDRESS
USTCH  =  13    ;NUMBER OF CHANNELS/TASKS
USTCT  =  14    ;CURRENTLY ACTIVE TCB
USTAC  =  15    ;START OF ACTIVE TCB CHAIN
USTFC  =  16    ;START OF FREE TCB CHAIN
```

USER STATUS TABLE OF LOADING INFORMATION (Continued)

```
USTIN   =   17          ;INITIAL START OF NREL CODE
USTOD   =   20          ;OVERLAY DIRECTORY ADDRESS
USTSV   =   21          ;FORTRAN STATE VARIABLE SAVE ROUTINE
USTEN   =   USTSV       ;LAST ENTRY
```

Location 400 - USTPC is the program counter. The loader initializes this word to 0, indicating that the program has never run.

Location 401 - USTZM points to the first available location in page zero for page zero relocatable code.

Location 402 and 403 - USTSS and USTES point to the start and end of the symbol table respectively as shown in the diagram on page 1-4. The loader sets 402 and 403 to 0 if the debugger is not loaded.

Location 404 - USTNM contains NMAX. The loader sets the pointer to the first free location for further loading or for allocation of temporary storage at run time.

Location 405 - USTSA points to the program starting address, specified by the .END statement. If no starting address is specified by any loaded program, -1 is stored in 405. If several programs specify starting addresses, USTSA contains the address specified in the last program loaded. (Location 377 contains a JMP @.+6, which transfers control to the program starting address. Therefore, the user can conveniently restart his program at 377, assuming that he has specified a starting address.)

Location 406 - USTDA points to the starting address of the debugger, or if the debugger is not loaded, 406 contains -1.

Location 407 - USTHU is set to the value of NMAX at the termination of loading. Since no operating system changes USTHU during program execution, it can be used to reset NMAX when a program is restarted.

Location 410 - USTCS contains the size of the FORTRAN unlabeled COMMON area, used when the binary relocatable programs being loaded contain .CSIZ blocks, such as those generated by the FORTRAN compiler.

Location 411 and 412 - USTIT and USTBR are set to 0 by the loader.

Locations 413-16, 420-21 - These locations are compatible with RDOS.

Location 417 - USTIN contains the address of the start of normally relocatable code (440_8).

END OF CHAPTER

CHAPTER 2

SOS MAGNETIC TAPE/CASSETTE RELOCATABLE LOADER

Under the Stand-alone Operating System, programs for systems that do not use magnetic tape or cassette I/O are loaded using the Stand-alone Relocatable Loader, 091-000038, as described in Chapter 1. Relocatable loader 091-000038 is supplied in absolute binary.

Programs for SOS systems that have either magnetic tape or cassette I/O, however, are loaded by the relocatable loader 089-000120, which is supplied as part of the SOS cassette or magnetic tape system.

OPERATION

The SOS relocatable loader 089-000120 must be loaded by the core image loader in accordance with the procedures outlined in the Stand-alone Operating System User's Manual.

Once loaded, the relocatable loader will print the following prompt at the teletypewriter:

RLDR

The user responds by typing a command line giving the names of files used as input to and output from the relocatable loader.

The user response to the RLDR prompt consists of a list of file names which may have local switches. The command causes the relocatable loader to produce from one or more .RB or .LB files, an executable core-resident program and a core image (save) file on magnetic tape or cassette. Both files start at address zero. The same file cannot be used for both input to and output from the relocatable loader. At least one input file and an output save file must be designated in the command line.

The SOS magnetic tape/cassette relocatable loader is compatible with the RDOS relocatable loader and builds a core resident program in much the same way:

> The user program ZREL code starts at location 50 and builds upwards in page zero.

OPERATION (Continued)

The User Status Table is contained in locations 400-437.

The User NREL code starts at location 440 and builds upward in memory.

The symbol table is retained in core only if the symbolic debugger, Debug III, is loaded. At termination of loading, the symbol table is moved down to the end of NREL code.

The maximum core size of each loaded program cannot exceed the maximum core address less 1325_8. The 1325 locations are required for the core image loader and pass 2 of the relocatable loader.

Upon completion of a successful load, the message "OK" is output on the teletype-writer and the system halts with the loaded program in core.

SYMBOL TABLE

The symbol table is built in high core and moved down to the end of NREL code at termination of loading. The symbol table is retained in core only if the symbolic debugger, Debug III is loaded. Debug III is supplied on relocatable binary tape 089-000073 and must be loaded as one of the input files in the RLDR command line if a symbol table is desired. The symbol table is similar to the one shown on page 1-7 for the Stand-alone Relocatable Loader.

USER STATUS TABLE

Locations 400-437 contain the User Status Table (UST). The table is given below:

```
USTPC  =  0
USTZM  =  1      ;ZMAX
USTSS  =  2      ;START OF SYMBOL TABLE
USTES  =  3      ;END OF SYMBOL TABLE
USTNM  =  4      ;NMAX
USTSA  =  5      ;STARTING ADDRESS
USTDA  =  6      ;DEBUGGER ADDRESS
USTHU  =  7      ;HIGHEST ADDRESS USED
USTCS  =  10     ;FORTRAN COMMON AREA SIZE
USTIT  =  11     ;INTERRUPT ADDRESS
USTBR  =  12     ;BREAK ADDRESS
USTCH  =  13     ;NUMBER OF CHANNELS/TASKS
USTCT  =  14     ;CURRENTLY ACTIVE TCB
USTAC  =  15     ;START OF ACTIVE TCB CHAIN
```

USER STATUS TABLE (Continued)

```
USTFC   =   16      ;START OF FREE TCB CHAIN
USTIN   =   17      ;INITIAL START OF NREL CODE
USTOD   =   20      ;OVERLAY DIRECTORY ADDRESS
USTSV   =   21      ;FORTRAN STATE VARIABLE SAVE ROUTINE
USTEN   =   USTSV   ;LAST ENTRY
```

COMMAND LINE

When the prompt RLDR is output, the user responds on the same line with a list of input and output file names. Switches may be attached to one or more of the file names, and each space is separated from the next by at least one blank space. The general format of the command line is:

filename$_1$...filename$_n$)

At a minimum, the command line must contain at least one input file name and one output save file name:

inputfilename outputfilename/S)

where: S is a switch indicating the save file.

A number of switches may be appended to the names of input and output files in the command line. They are as follows:

Output File Name Switches

/S The /S follows the name of a cassette or magnetic tape file, indicating that that device will be used for output of the save file. If no save file is specified or if a file is incorrectly specified as a save file, an error message will result and the loader will reinitialize itself, printing the prompt " RLDR".

/L The /L follows the name of a device and causes a numerically ordered listing of the symbol table to the device. The output device for the listing cannot be the same as that used for the save file.

/A This switch may be appended to the same device as that having the /L

Output File Name Switches (Continued)

switch. It causes an alphabetic as well as numeric listing to result. The /L switch must be present.

Input File Name Switches

/N NMAX, the starting address for loading a given input file may be changed from the default address by use of this switch. The /N follows an absolute address, given in octal, and precedes the name of the input file to be loaded beginning at the octal address. The octal address given must be greater than the current value of NMAX.

/P Files to be loaded may be on different cassettes. /P following a file name causes a halt before the file of that name is loaded that allows the user to mount a new cassette containing the input file. When the loader halts, the message: PAUSE - NEXT FILE filename is printed, where filename is the name of the file that had the /P switch. When the new cassette is mounted, the user restarts loading by pressing any teletype-writer key, e.g., RETURN.

/U /U causes local user symbols appearing within the file preceding the switch to be loaded.

/n n is a digit in the range 2-9. The input file preceding the switch is loaded the number of times specified by the switch.

Command Line Error Messages

Following are the possible command line error messages:

NO INPUT FILE SPECIFIED

NO SAVE FILE SPECIFIED

SAVE FILE IS READ/WRITE PROTECTED

The save file device must be either a cassette or magnetic tape and must permit both reading and writing.

aaaaa I/O ERROR nn

where: aaaaa is the address associated with the error.

Command Line Error Messages (Continued)

nn is one of the following RDOS codes:

1 - Illegal file name
7 - Attempt to read a read-protected file.
10 - Attempt to write a write-protected file.
12 - Non-existent file.

Examples of Command Lines

$TTO/L/A CT2:0/S $PTR CT1:6 16500/N CT1:0)

Input files are the $PTR, CT1:6 and CT1:0. NMAX is reset for CT1:0 to 16500_8. The save file is written to CT2:0 and an alphabetically ordered listing is output on the teletypewriter.

If one of the input files, CT1:6, CT1:0 or the $PTR contains the debugger, a symbol table will be generated.

MT1:0/S MT0:1 MT0:2 $PTP/L)

Input files are MT0:1 and MT0:2. The save file is output to MT1:0. A numeric listing is to the paper tape punch.

CT0:0/S CT1:2 CT1:0/P)

Input files are on different cassettes, so the /P switch allows a pause for the user to change the cassette tape on unit 1. The save file is output to CT0:0.

RESTART PROCEDURE

The loader can be stopped and restarted at location 377 any time in Pass 1 (up until the end of the listing of the symbol table). Once Pass 2 starts, the loader must be reloaded from cassette or magnetic tape.

ERROR MESSAGES

In addition to the command line error messages described on the previous page, the loader produces explicit error messages that are printed to the console. These include both fatal and non-fatal error messages. The error messages are followed by an appropriate identifying location, symbol, or both.

ERROR MESSAGES (Continued)

Non-fatal Errors

Non-fatal errors do not stop loading but may change the intended state of the output file. The non-fatal error messages are:

DISPLACEMENT OVERFLOW nnnnnn

A displacement overflow error occurs if the loader finds the displacement is too large when attempting to resolve an external displacement. The displacement is too large if:

the index = 00 and the unsigned displacement is > 377.

the index ≠ 00 and the displacement is not in the range:
-200 ≤ displacement < +200

nnnnnn is the absolute address where overflow occurred. The displacement is left unresolved with a value of 000.

ILLEGAL BLOCK TYPE nnnnnn

The error message normally occurs if the input file is not a relocatable binary or library file. The file in error will not be loaded. Octal number nnnnnn is the block code of the illegal block.

MULTIPLY DEFINED ENTRY sssss nnnnnn

This error occurs when an entry symbol or named common (.COMM) symbol, sssss, having the same name as one already defined is encountered during loading. nnnnnn is the absolute address at which the symbol was originally defined.

EXTERNAL UNDEFINED IN EXTERNAL EXPRESSION sssss

This error occurs if a .GADD block is encountered that references an as yet undefined symbol, sssss. Zero is stored in the memory cell.

BINARY WITHOUT END BLOCK

This error occurs when a binary file has no end block. The file is loaded up to the point where the error is discovered.

Non-fatal Errors

ILLEGAL NMAX VALUE nnnnnn

This error occurs when the user attempts to force the value of NMAX to a value lower than the current value of NMAX, i.e., if the octal value following a /N local switch is lower than the current value of NMAX. nnnnnn is the illegal value. NMAX is unchanged.

NO STARTING ADDRESS FOR LOAD MODULE

This error occurs if at assembly time the user failed to terminate at least one of the programs to be loaded with a .END pseudo-op that was followed by a starting address for the save file. The starting address can be patched by the user into location 405 (USTSA) of the User Status Table.

EXTERNAL NORMAL/DISPLACEMENT CONFLICT sssss

This error occurs when a symbol sssss appears in a .EXTD block in one module to be loaded and in a .EXTN block in another module.

CAUTION OLD ASSEMBLY ssss

This error occurs when this program was assembled by an incompatible assembler.

Fatal Errors

If an error is fatal, the error message and the location at which it was discovered are followed on the next line by a second message:

FATAL LOAD ERROR

and the loader gives the prompt, RLDR. For example:

```
CHECKSUM ERROR nnnnnn
**FATAL LOAD ERROR**
RLDR
```

The fatal errors are:

CHECKSUM ERROR nnnnnn

This error occurs if a checksum that is computed on some block differs from zero. nnnnnn is the incorrect checksum.

Fatal Errors (Continued)

NEGATIVE ADDRESS nnnnnn

This error occurs if bit 0 of an address word is set to 1. The assembler restricts addresses to the range: $0 \leq \text{address} < 2^{15}$; however, the error can be caused by a reader error. nnnnnn represents the negative address.

PAGE ZERO OVERFLOW nnnnnn

This error occurs in loading page zero relocatable data if the data overflows the page zero boundary (377_8). The absolute address of the first word of the data block that caused the overflow is given by nnnnnn.

NAMED COMMON ERROR sssss nnnnnn

This error occurs if two programs have different sizes for a given area of labeled COMMON (defined by .COMM statements), or if the symbol table flags that are associated with the symbol are not 00010_2. sssss gives the symbol name of the labeled COMMON and nnnnnn indicates the size of the labeled COMMON requested by the present .COMM .

SYMBOL TABLE OVERFLOW

This error occurs during loading if the size of the symbol table becomes so large that it would overwrite the loader in core.

EXTERNAL LOCATION UNDEFINED sssss

This error occurs if a .GLOC block is encountered with data to be loaded at the address of a symbol, sssss, that is as yet defined.

MEMORY OVERFLOW nnnnnn

If the value of NMAX plus the loader size itself conflicts with the bottom of the loader's symbol table, a memory overflow error occurs. The error implies that the user programs are too large to be loaded in the memory configuration. The octal number nnnnnn is the value of NMAX that caused the overflow.

END OF CHAPTER

CHAPTER 3

RDOS RELOCATABLE LOADER

GENERAL

The RDOS Relocatable Loader is used to build on disk, a Save file of an executable program. The Save file is built from relocatable binary files, and system libraries.

The loader is invoked through the RDOS CLI command, RLDR. The order in which the Save file is built is the order in which the names of programs and libraries appear in the RLDR command line. Each library is searched once each time its name appears in the command line.

COMMAND LINE

The general formats of the RLDR command line are:

RLDR $rtname_1$ ... $rtname_N$ [$ovname_1$ ...$ovname_N$]

RLDR $rtname_1$ [$ovname_1$ ... $ovname_N$] $rtname_N$

where *rtname* is the name of a relocatable binary or library file and *ovname* is the name of a relocatable binary file. The square brackets enclose relocatable binarys to be included in overlays.

Overlays can be built anywhere in the Save file. Overlays are built in nodes, one node for each set of overlay names enclosed in a pair of square brackets.

Each overlay within a node is made up of one or more binary files. Commas separate the binary files for the different overlays. For example:

[$ovname_1$ $ovname_2$, $ovname_3$, $ovname_4$]

describes a node with three overlays. The first overlay is made up of two binary files, $ovname_1$ and $ovname_2$. The second and third overlays are made up of one relocatable binary each $ovname_2$ and $ovname_3$ respectively. The size of a node is determined by the size of the largest overlay within it. Space for overlays is allocated in units of 256 decimal words (with the exception of virtual overlays

COMMAND LINE (Continued)

which are allocated in units of 1024 decimal words). Thus if binary files *ovname*$_1$, *ovname*$_2$, *ovname*$_3$ and *ovname*$_4$ consist of 10, 15, 20 and 40 words respectively, the last overlay will determine the node size. Since space is allocated 256 words at a time, the node will be allocated 256 words. If the sizes were 200, 60, 252 and 121 words instead, the first overlay with a total of 260 words would determine node size. Two units of 256 words or a total of 512 words would be required.

If the system library (SYS. LB) is not specifically named as a *rtname* in the command line, it is automatically searched after the last specified file name.

COMMAND LINE SWITCHES

Both global and local switches are provided to control the building of the Save file and the output of the loading operation. Global switches are appended to the command name (RLDR) and local switches are appended to file names or to octal values specified in the command line.

Global Switches

The global switches are /N - /A, /E, /H, and /M - /D and /S - /C and /Z - /X and /Y, /I, /K, /O, /P, /U and /B.

/N — If the command line does not specifically name the system library (SYS. LB), it is automatically searched after the last named file in the command line. To prevent this search, /N is appended to the command RLDR.

/A, /E, /H, and /M — When /A is appended to RLDR and local switch /L is appended to an output device name, an alphabetically ordered listing of the symbol table is produced, along with the numerically ordered listing. For example:

```
RLDR /A  ROOT  $LPT/L
```

produces both an alphabetically ordered and a numerically ordered listing of the symbol table on the line printer.

NOTE: $LPT/L can be used without using /A, and only the numerically ordered listing is produced.

/A, /E, /H, and /M (Continued)

If a listing file is not specified in the command line, error messages are output on the system console. However, when a listing file is specified, error messages to the console are suppressed; being output on the listing device. To cause the error messages to be output to the system console when a listing device is used, append /E to the command RLDR. Thus,

```
RLDR  ROOT  $LPT/L
```

causes error messages to be output only to the line printer, while:

```
RLDR/E  ROOT  $LPT/L
```

causes error messages to be output on the system console as well as on the line printer.

Normally, numbers are output in octal. To have numbers output in hexadecimal, append /H to the command RLDR.

To shorten the time it takes to create a Save file in systems using a teleprinter system console, append /M to the command RLDR. This inhibits all output to the system console.

NOTE: /M should be used with caution since error messages also are suppressed.

/D and /S

When /D is appended to RLDR, the symbolic debugger DEBUG III is incorporated in the Save file. To have IDEB become part of the Save file rather than DEBUG III, the name IDEB.RB must appear in the command line. Hence:

```
RLDR/D  ROOT
```

causes DEBUG III to be loaded, and

```
RLDR/D  ROOT  IDEB.RB
```

causes IDEB to be loaded.

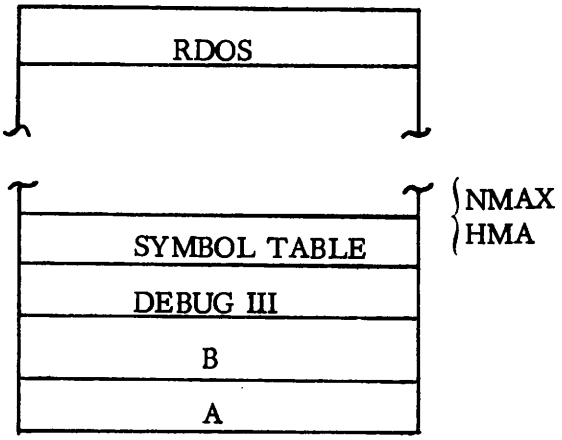
The symbol table becomes part of the Save file only when /D is specified or if the symbol ".SYM." is included in one of the loaded files.

/D and /S
(Continued)

When the command

RLDR /D A B

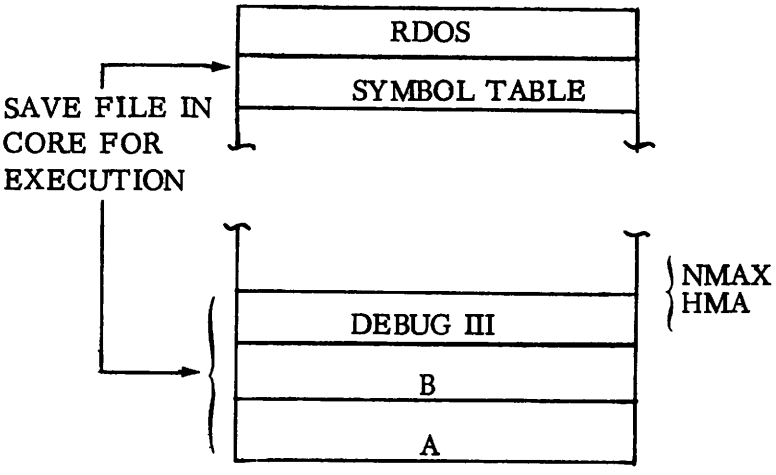
is executed, a Save file will be created which will have the following configuration when called into core.



When the command line

RLDR /D/S A B

is executed, the Save file created looks like the following when called into core.



When the global /S switch is used, the symbol table is built into the save file immediately under the RDOS system or at

/D and /S (Continued)

the top of the address space in a mapped system. Otherwise, it occupies the locations immediately above the highest (last) relocatable binary file.

Note the relative positions of NMAX and HMA in the preceding two diagrams. The area of free core, while not changed in size, has different starting and ending locations depending on whether /S is used when the Save file is created.

/C and /Z

Switches /C and /Z are used to create a Save file in the RDOS environment that can be executed in the SOS or RTOS environment or in an environment without an executive.

NOTE: Save files created in RDOS using /C or /Z in the RLDR command line cannot be executed in the RDOS environment.

When /C is appended to RLDR

- Initial NMAX is set at 440_8.
- Save file starts at absolute location 0.
- USTSA contains the program starting address.
- SYS.LB is not searched unless it appears in the command line.

The Save file created using /C can be executed in the SOS or RTOS environment.

When /Z is appended to RLDR, the Save file begins at absolute location 0 so the routine can use locations 0 - 15. The Save file must be output using the CLI command MKABS /Z to produce an absolute binary file that can be loaded for execution with the stand-alone loader or with HIPBOOT.

/X and /Y

The switches /X and /Y are provided to perform a system load. The /X switch allows up to 128_{10} system overlays and /Y allows up to 256_{10} system overlays. It should be noted that /X and /Y are mutually exclusive switches.

/I — The /I switch allows loading into locations that would otherwise be illegal e.g., $0\text{-}15_8$. /I also suppresses loading of a UST. /I is used with /Z.

/K — Do not delete the symbol file at the end of the load.

/Q — Do not put symbols in the same file even though /D is used.

/P — Print the starting relative address of a module along side its title as it is loaded.

/U — Do not resolve undefined symbols to -1.

/B — Force short TCB's.

Case 1. No COMM task, no command line declaration.

Result. 1 long TCB (independent of /B).

Case 2. Either a COMM Task or a command line declaration.

Result. Using /B switch causes all short TCB's. No /B causes all long TCB's.

Local Switches

/C and /K — Switch /C is inserted in the command line following an octal value. The octal value represents the number of channels to be assigned when the Save file is brought into core for execution. Unless specifically included in the command line or specified in a .COMM TASK statement in the first relocatable binary, 8_{10} channels are assigned by default. For example, if 9_{10} channels are required when the Save file is executed, the command line should include:

```
RLDR ROOT1 11/C
```

Local Switches (Continued)

/C and /K (Continued)

Switch /K also is inserted in the command line following an octal value. The octal value represents the number of required tasks. Unless specifically included in the command line, 1 task is assigned. Therefore, if ROOT1 in the preceding example is a multi-task program with 8_{10} tasks, then the command line should be:

```
RLDR  ROOT1  11/C  10/K
```

NOTE: The octal values specified by /C and /K replace the contents of USTCH in the User Status Table. The number of channels is stored in the right-hand byte and the number of tasks in the left-hand byte. These values replace the corresponding values in a .COMM TASK block. Note also that in a multi-task program, either a .COMM TASK statement must appear or the RLDR command line must contain both /C and /K switches, otherwise the fatal error LOAD OVERWRITE 17 occurs.

/E and /L

Recall from the discussion of global switch /E that error messages can be output to the system console even though a listing file is specified. Local switch /E is used to name a device other than the system console to which error messages are to be output when global switch /E and local switch /L are specified in the same command line:

```
RLDR/E  ROOT1  $LPT/L  $TTO1/E
```

/E and /L (Continued)

causes a list of the symbol table including error messages to be produced on the line printer and only the core map portion of the load listing and error messages to be output on the teleprinter. No output goes to the system console even though the global switch /E is present.

Local switch /L is used to designate an output device to receive the symbol table when it is desired to save the symbol table. Recall that the symbol table is output numerically ordered unless global switch /A also appears in the command line. In the latter case, both a numerically and an alphabetically ordered list is produced.

/F and /Z

By default, the Save file is built to be executed in the background. Local switch /F can be included in the command line to cause the save file to be built for execution in the foreground. The switch follows an octal value specifying the desired foreground NREL starting address. If the value specified is not an integer multiple of 400 plus 16, the loader rounds it up to the next highest such multiple. Therefore, if

```
RLDR  ROOT1  1421/F
```

is specified, the loader sets the foreground NREL to 2016.

Normally, ZREL is set at location 50 by default. To change the ZREL starting location, include an octal value followed by switch /Z in the command line. This switch also is used to specify the ZREL foreground position. Thus, the command line

```
RLDR  ROOT1  70/Z  1421/F
```

sets the foreground ZREL at location 70 and the foreground NREL at 2016.

/S and /U

When the Save file is created the extension .SV is appended to the first named file in the command line, and that name with the extension is given to the Save file. If overlays are included in the Save file, the overlay file also is named for the first specified file and has the extension .OL. Thus, the names of the Save file and overlay file from the command

```
RLDR  ROOT1  [R1, R2]
```

/S and /U (Continued)

are ROOT1.SV and ROOT1.OL. To have the Save and overlay files named something different, append the switch /S to any file name in the command line.

```
RLDR  ROOT1  ROOT/S  [R1, R2]
```

causes the Save file to be named ROOT.SV and the overlay file to be named ROOT.OL.

Local symbols (i.e. not declared as .ENT's) are not normally included in the symbol table. To have them incorporated in the symbol table, when debugging the program for example, append local switch /U to the name of the relocatable binary file containing the program to be debugged. Remember, to have the symbolic debugger incorporated in the Save file, global switch /D must be appended to the command RLDR. For example,

```
RLDR/D  ROOT1/U  ROOT2
```

causes DEBUG III to be incorporated in the Save file and the user defined symbols in the program file ROOT1 to be included in the symbol table along with the external symbols from ROOT1 and ROOT2. ROOT1 must have been assembled with global /U switch.

/N

When the Save file is loaded for execution, it is loaded into core upwards from the lowest available address. To specify where a particular program in the Save file is to start in core, enter the octal value of the desired location followed by switch /N followed by the relocatable binary file name. The value specified becomes the program's NMAX and must be larger than the current NMAX. Thus, the command

```
RLDR  ROOT1  2000/N  ROOT2
```

causes the program in binary file ROOT2 to be loaded into core starting at location 2000. The number specified cannot be lower than any previously loaded address.

/V

Local switch /V only applies to operations in the mapped environment. It enables the creation of a virtual overlay. To create a virtual overlay, the switch /V immediately follows the right-hand bracket of an overlay specification in the line. For example,

```
RLDR  ROOT1  [R1, R2]/V
```

When a virtual overlay is used it can appear anywhere in the command line but all virtual overlays must precede all non-virtual overlays. Thus,

```
RLDR  ROOT1  [R1]/V  [R2]  [R3]/V
```

produces an error; but,

```
RLDR  ROOT1  [R1]/V  [R3]/V  [R2]
```

does not.

SYMBOL FILE

RLDR maintains its symbol table in a file named <save filename>.ST. This file is deleted at the end of a load unless the global /K switch is used. A table containing undefined symbols is maintained in core. This table grows as external symbols are encountered and then shrinks as they are resolved. A " Symbol Table Overflow" error means that this table has grown too large for the area from the top of RLDR to the system (on a foreground program). Since the maximum size of this table is a function of the peak number of undefined symbols rather than the total number of symbols in the load, rearranging the order in which files are loaded may avoid the error.

SIZE OF THE SAVE FILE

Since relocatable binaries are loaded directly to a save file on disk, it is possible to create a save file that is too large to be executed within the core limitations of the machine that performed the loading. While there is no direct method by which the user can avoid this possibility, there is a way to determine whether his save file will run in available core once it is loaded.

If the user appends the D and S switches to RLDR, the symbol table will be fixed at the highest location available in the machine in which loading occurs. The symbol table will be appended to the save file at these locations. If the symbol table is successfully appended to the save file without a resulting fatal error, the user is assured that the loaded programs will fit into his current core configuration.

Should the current NMAX of the save file be higher than the last location in the symbol table when an attempt is made to append the symbol table, the fatal error message:

SYMBOL TABLE TOO LARGE FOR CORE STORAGE

will be given. Note that the occurrence of the error does not necessarily mean that the loaded programs cannot fit in current core, since the debugger was loaded and the symbol table requires space. However, absence of the error message insures that the save file will fit into available core.

.LMIT FEATURES

When only part of a module is loaded, through the use of .LMIT pseudo-op entry, symbols defined in the nonloaded part of the module are treated in a special way by RLDR. RLDR determines whether a symbol is in the loaded part of the module by comparing its value to the limit value. If the value is greater than the value of the limited symbol, the entry symbol is assumed to be defined in the non-loaded part of the file. Entry symbols defined in the non-loaded part of the module are treated in a way to show that the code they represent does not really exist in the same file. If the symbol has previously been added to the symbol file by a module that declared it as an EXTD or an EXTN, the value of the symbol is set to -1. If the symbol has not been declared by a previous module, it is not added to the symbol table at all. The order in which entry symbols appear in the RB is important in this process. RLDR can only make this determination after it encounters the limited symbol. Thus only entry symbols following the limited symbol are defined as described above. The order in which entry symbols appear can be controlled when MAC is used to create RB's. MAC outputs entry symbols in the opposite order of the way they are declared with .ENT pseudo-ops.

LOAD LISTING

The load listing has two distinct parts: a core map and a symbol list.

Core Map

```
00      ROOT0
01              001470
02              000,000  OL00  000004
03              000,001  OL01  000004
04              002070
05      ROOT1
06              002276
07              001,000  OL10  001010
08              003676
09       TMIN
10      NO STARTING ADDRESS FOR LOAD MODULE
11           NMAX  003752
12           ZMAX  000050
13           CSZE  000000
14             EST  000000
15             SST  000000
```

NOTE: Line numbers do not appear on the core map. They are included here only for discussion purposes.

Lines 00, 05, and 09 contain root titles.

Titles of modules loaded from libraries are also printed. Each title is printed when the corresponding START block is encountered, except in case of error, in which case the title is printed before the corresponding error message.

LOAD LISTING (Continued)

Load map flags are:

XD	-	external displacement undefined
XN	-	external normal undefined
C	-	named common
M	-	multiply defined

A named common symbol is followed by both its address and size.

Lines 01 and 06 contain node points, which are the starting addresses of the overlay areas.

Lines 04 and 08 contain overlay area ending addresses.

Lines 02, 03, and 07 contain node and overlay numbers, titles of modules loaded in the overlays, and the length in words of the overlays. For example, line 03 is interpreted as follows

000,001		OL01	000004
node number	overlay number	Title of overlay	Length of overlay in words

Notice in the sample above that node 0 begins at location 001470 and ends at location 002070, an area of 400_8 words even though the overlay OL00 and OL01 are each only 4 words long. Notice also that node 1 is allocated 1400_8 words but that overlay 0, named OL10, is only 1010_8 words long. In both of these cases, the loader rounded up the size of the node to the next integer multiple of 400_8.

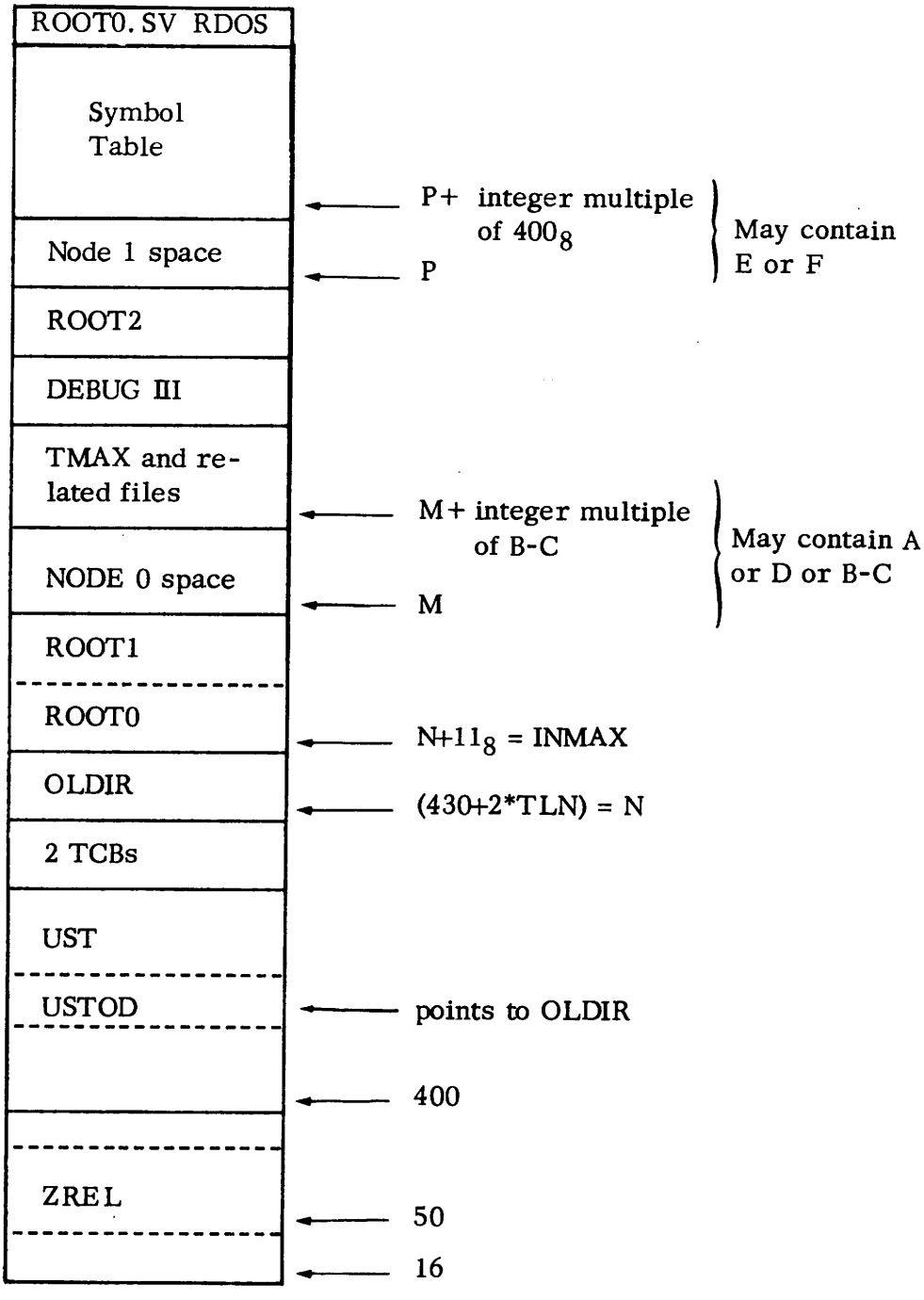
NOTE: Symbol USTAD always appears on the Symbol list and indicates the starting address of the User Status Table.

EXAMPLES

Following are some examples of command lines, shown with the save and overlay files that are created.

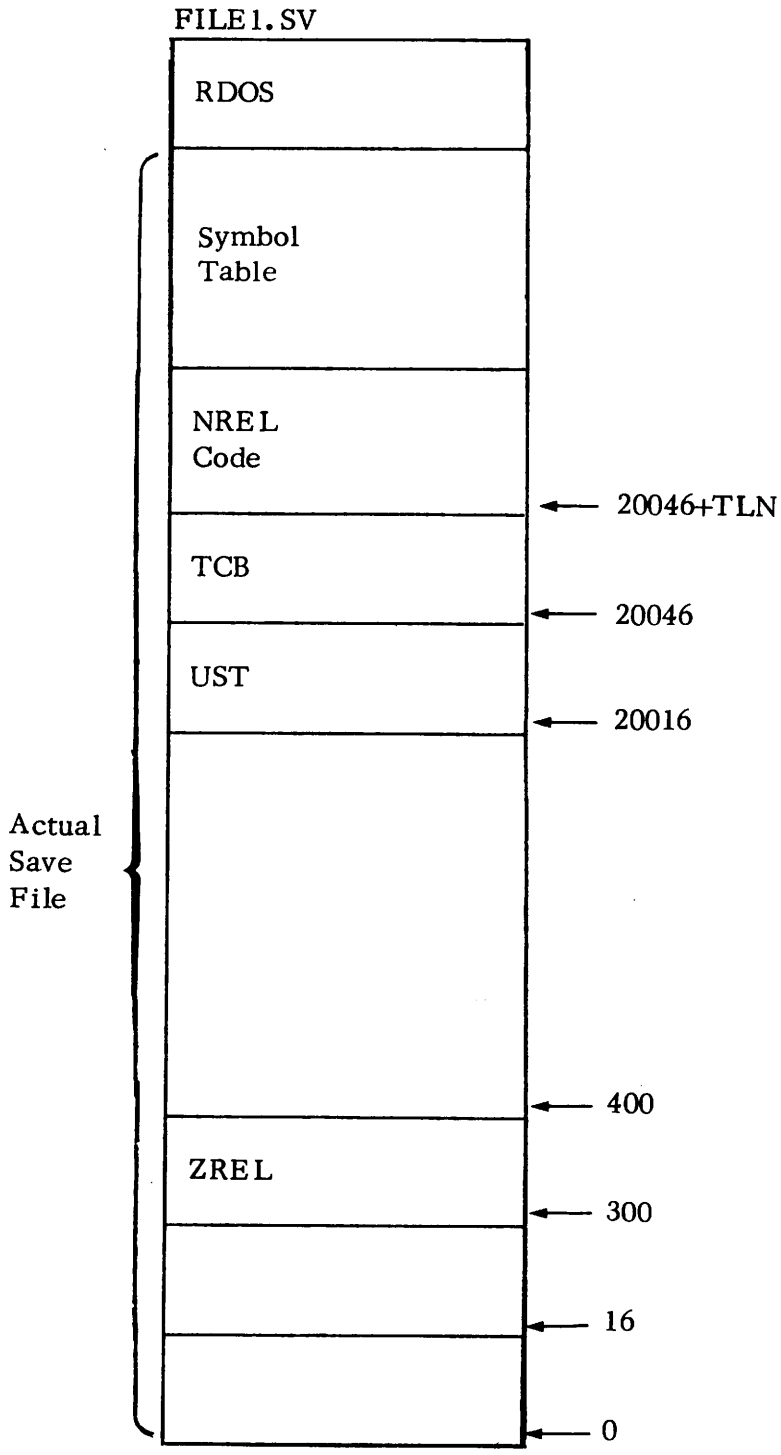
EXAMPLE (Continued)

RLDR ROOT0 ROOT1 [A, B, C, D] SYS.LB ROOT2 [E, F] 2/K)



EXAMPLES (Continued)

```
RLDR FILE1 300/Z 20000/F FILE2 )
```



ERROR MESSAGES

Explicit error messages are output by the loader for both non-fatal and fatal errors. The error messages are output to the file designated by the appropriate RLDR command line switches (local /E and /L and global /E) and are followed by an appropriate identifying location, symbol or both.

Non-fatal Errors

Non-fatal errors do not stop loading but may change the intended state of the user's save file. The non-fatal error messages are:

DISPLACEMENT OVERFLOW nnnnnn

A displacement overflow error occurs if the loader finds the displacement is too large when attempting to resolve an external displacement. The displacement is too large if:

the index = 00 and the unsigned displacement is $>$ 377.

the index $\neq$ 00 and the displacement is outside the range:

$$-200 \leq \text{displacement} \leq +200$$

nnnnnn is the absolute address where overflow occurred. The displacement is left unresolved with a value of 000.

ILLEGAL BLOCK TYPE nnnnnn

The error message normally occurs if the input file is not a relocatable binary or library file. The file in error will not be loaded. Octal number nnnnnn is the illegal block.

MULTIPLY DEFINED ENTRY sssss nnnnnn

This error occurs when an entry symbol or named common (. COMM) symbol, sssss, having the same name as one already defined, is encountered during loading. nnnnnn is the absolute address at which the symbol was originally defined. Of course, two or more named commons with the same name can occur, but an attempt to redefine an ENT as a COMM or a COMM as an.ENT will result in an error.

ERROR MESSAGES (Continued)

EXTERNAL UNDEFINED IN EXTERNAL EXPRESSION sssss

This error occurs if a .GADD block is encountered that references an as yet undefined symbol, sssss. Zero is stored in the memory cell.

ILLEGAL NMAX VALUE nnnnnn

This error occurs when the user attempts to force the value of NMAX to a value lower than the current value of NMAX, i.e., if the octal value following a /N local switch is lower than the current value of NMAX. nnnnnn is the illegal value. NMAX is unchanged.

SYSTEM LIBRARY NOT FOUND

This error occurs when the system library (SYS.LB) could not be found on the current directory.

NO STARTING ADDRESS FOR LOAD MODULE

This error occurs if at assembly time the user failed to terminate at least one of the programs to be loaded with a .END pseudo-op that was followed by a starting address for the save file. The starting address can be patched into the TCBPC word of the TCB pointed to be USTCT. It must be stored as the starting address multiplied by 2.

BINARY WITHOUT END BLOCK

The error occurs when a binary file has no end block. The file is loaded up to the point where the error is discovered.

TASKS OR CHANNELS SPECIFIED = 0

This error occurs when there was a .COMM task block with left or right byte of its equivalence word = 0 or when 0/K or 0/C appears in the COM.CM file.

ERROR MESSAGES (Continued)

Non-Fatal Errors (Continued)

NO SCHEDULER STARTING ADDRESS

In a stand-alone load (global /C) this error occurs if no start block contained a starting address. The starting address can be patched into USTSA.

WARNING *** ZERO LENGTH OVERLAY

This error indicates that an attempt has been made to load an overlay that contains nothing.

Fatal Errors

If an error is fatal, the error message and the location at which it was discovered are followed on the next line by a second message:

FATAL LOAD ERROR

For example:

```
LOAD OVERWRITE 001700
**FATAL LOAD ERROR**
```

The message is output to the error file, and return is made to the CLI which prints the message:

FATAL SYSTEM UTILITY ERROR

The fatal errors are:

CHECKSUM ERROR nnnnnn

This error occurs if a checksum that is computed on some block differs from zero. nnnnnn is the incorrect checksum.

NEGATIVE ADDRESS nnnnnn

This error occurs if bit 0 of an address word is set to 1. The assembler restricts addresses to the range: $0 \leq \text{address} < 2^{15}$: however, the error can be caused by a reader error. nnnnnn represents the negative address.

ERROR MESSAGES (Continued)

Fatal Errors (Continued)

NAMED COMMON ERROR sssss nnnnnn

This error occurs if two programs have different sizes for a given area of labeled COMMON (defined by .COMM statements) and the second is larger. sssss gives the symbol name of the labeled COMMON and nnnnnn indicates the size of labeled COMMON requested by the present .COMM.

LOAD OVERWRITE nnnnnn

The loader does not permit save or overlay file locations to be overwritten by subsequent data once they are loaded. If an attempt to overwrite is made, this error occurs. The absolute address where the overwrite was attempted is given by nnnnnn.

EXTERNAL LOCATION UNDEFINED sssss

This error occurs if a .GLOC block is encountered with data to be loaded at the address of a symbol, sssss, that is as yet undefined.

PAGE ZERO OVERFLOW nnnnnn

This error occurs in loading page zero relocatable data if the data overflows the page zero boundary (377_8). The absolute address of the first word of the data block that caused the overflow is given by nnnnnn.

SYMBOL TABLE TOO LARGE FOR CORE IMAGE

This error occurs when a global switch /S has been given in the RLDR command line and the symbol table would overwrite loaded programs in the save file built by the loader.

ILLEGAL LOAD ADDRESS

This error occurs when an attempt is made to load into locations 0-15.

Fatal Errors (Continued)

SYMBOL TABLE OVERFLOW

This error occurs during loading if the size of the symbol table becomes so large that it would overwrite the loader in core.

RDOS ERROR

This error indicates that the loader issued a system call that could not be completed and that resulted in an exceptional return. See the RDOS User's Manual for system calls and possible error returns.

TASK MONITOR ERROR (USTCH)

This error occurs when a .COMM block with symbol TASK is encountered at a point when NMAX differs from the initial value of NMAX. This occurs if .COMM TASK occurs in some module after the first module is loaded.

OVERLAY DIRECTORY OVERFLOW

This error occurs when the number of nodes exceeds 128 or number of overlays at a given node exceeds 256.

BLOCK FORMATS OF RELOCATABLE BINARY TAPES

The order of blocks for input to the relocatable loader was shown on page 1-3 of the manual. Following are the formats for each type of block.

Relocatable Data Block

	word
2	1
word count	2
relocation flags 1	3
relocation flags 2	4
relocation flags 3	5
checksum	6
address	7
data	8
data	9
⋮	⋮
data	word count + 6

Entry Block*

	word
3	1
word count	2
relocation flags 1	3
relocation flags 2	4
relocation flags 3	5
checksum	6
symbol in radix 50 / flags	7
equivalence	8
⋮	9
symbol in radix 50 / flags	⋮
equivalence	word count +6

External Displacement Block

	word
4	1
word count	2
6	3
6	4
6	5
checksum	6
symbol in radix 50 / flags	7
77777	8
⋮	9
symbol in radix 50 / flags	⋮
7777	word count +6

* Entry blocks can contain either .ENT or .ENTO, which are differentiated by the contents of flag bits 11-15 of S1 (word 8, etc.)

Normal External Block

	word
5	1
word count	2
relocation flags 1	3
relocation flags 2	4
checksum	5
symbol in radix 50 / flags	6
address of last ref.	7
⋮	8
symbol in radix 50 / flags	⋮
address of last ref.	word count +6

Start Block

	word
6	1
word count (-1)	2
relocation flags 1	3
0	4
0	5
checksum	6
address	7

Title Block

	word
7	1
word count (-3)	2
0	3
0	4
0	5
checksum	6
title in radix 50 / flags	7
equivalence	8
	9

Local Symbol Block

	word
10	1
word count	2
relocation flags 1	3
relocation flags 2	4
relocation flags 3	5
checksum	6
symbol in radix 50 / flags	7
equivalence	8
⋮	9
symbol in radix 50 / flags	⋮
equivalence	word count +6

RDOS STATUS INFORMATION

The figure on page 3-14 shows that the start of user NREL code is given as variable location INMAX. (This value is given in USTIN of the UST.) The location is variable because the number of words required for RDOS status information: the overlay directory area, the task control block area, and the user status table area.

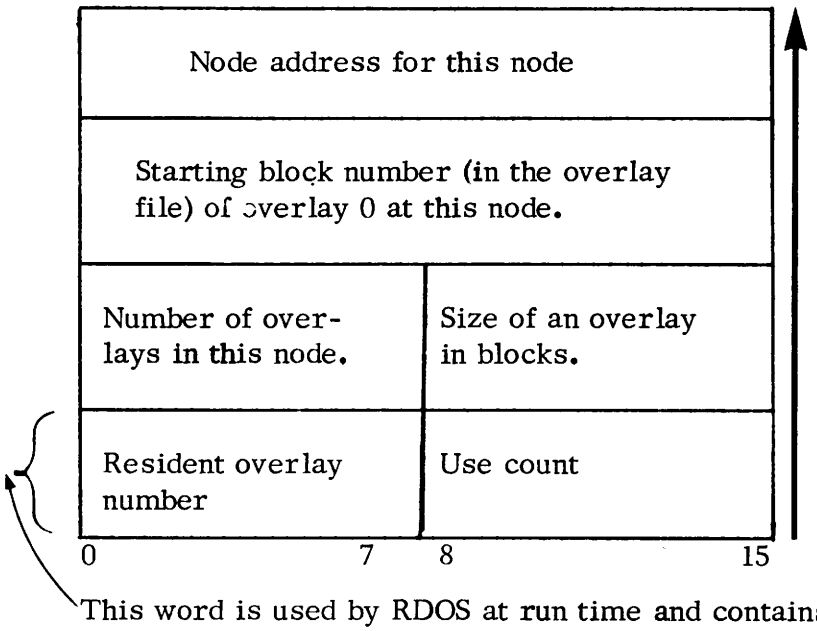
Overlay Directory Area

The overlay directory, OLDIR, is always created as part of the RDOS status information if there exists at least one overlay. If there are no overlays, no locations are set aside for OLDIR.

As described in the RDOS User's Manual, an overlayed program has a root or core-resident portion with a number of node points at which overlays may begin. The size of the overlay directory, OLDIR, will vary with the number of nodes, where up to 128_{10} nodes are permitted, and up to 256_{10} overlays are permitted in a given node. The number of words required for OLDIR is:

4*(number of nodes)+1

Word zero of OLDIR contains the count of number of nodes. The zeroth word is pointed to by USTOD in the User Status Table. For each node, the contents of the four words are as given in the diagram below.



Overlay Directory Area (Continued)

The overlay directory area is built up in the save file immediately above the Task Control Block area.

Task Control Block Area

The TCB (task control block) area of the RDOS status information contains information for each task in the program. The number of words required for the TCB area is obtained by multiplying the number of tasks by TLN.

(number of tasks) * (TLN)

TLN is the number of words required for each TCB and is defined on the user parameter tape, 090-000883. A listing of the parameter tape is found in the RDOS User's Manual.

The task control block area is built up in the save file immediately above the User Status Table.

User Status Table (UST)

A User Status Table, which is defined on the user parameter tape, 090-000883, is shown following.

```
USTPC = 0
USTZM = 1       ;ZMAX
USTSS = 2       ;START OF SYMBOL TABLE
USTES = 3       ;END OF SYMBOL TABLE
USTNM = 4       ;NMAX
USTSA = 5       ;STARTING ADDRESS OF TASK SCHEDULER
USTDA = 6       ;DEBUGGER ADDRESS
USTHU = 7       ;HIGHEST ADDRESS USED
USTCS = 10      ;FORTRAN COMMON AREA SIZE
USTIT = 11      ;INTERRUPT ADDRESS
USTBR = 12      ;BREAK ADDRESS
USTCH = 13      ;NUMBER OF CHANNELS
USTCT = 14      ;CURRENTLY ACTIVE TCB
```

User Status Table (UST) (Continued)

```
USTAC = 15        ;START OF ACTIVE TCB CHAIN
USTFC = 16        ;START OF FREE TCB CHAIN
USTIN = 17        ;INITIAL START OF NREL CODE
USTOD = 20        ;OVERLAY DIRECTORY ADDRESS
USTSV = 21        ;FORTRAN STATE VARIABLE SAVE ROUTINE
USTEN = USTSV     ;LAST ENTRY
```

USTPC is maintained by the system to provide compatibility with SOS, the Stand-alone Operating System.

USTZM contains ZMAX, the first location in page zero for page zero relocatable code.

Locations 402 and 403, USTSS, and USTES, respectively, point to the start and the end of the symbol table. Under default conditions, the loader loads the symbol table at the termination of loading so that the last location in the symbol table + 1 coincides with the value of NMAX after all programs are loaded. USTSS, USTES, and NMAX are updated. If the user requests that the symbol table be placed in upper core (/S switch on an RLDR command), the contents of 402, 403, and NMAX remain true and are not updated at the end of loading. If the debugger has not been loaded, locations 402 and 403 are set to zeroes.

USTNM contains the current value of NMAX at run time. This value changes as NMAX is increased or decreased.

Location 407, USTHU, is initialized by the loader to the value of NMAX at the termination of loading. This word is never changed by the operating system during program execution. It is used to reset USTNM whenever a program is started by the system.

USTIT is the interrupt address (CTRL A). At the termination of loading, this address is set to a -1. If unchanged at run time an unconditional return to the CLI occurs when a CTRL A interrupt occurs. The user core image is not saved. The user program can set USTIT at execution time to an address to which control will be transferred if a CTRL A interrupt occurs. The ACs will be lost upon transfer to such an address.

User Status Table (Continued)

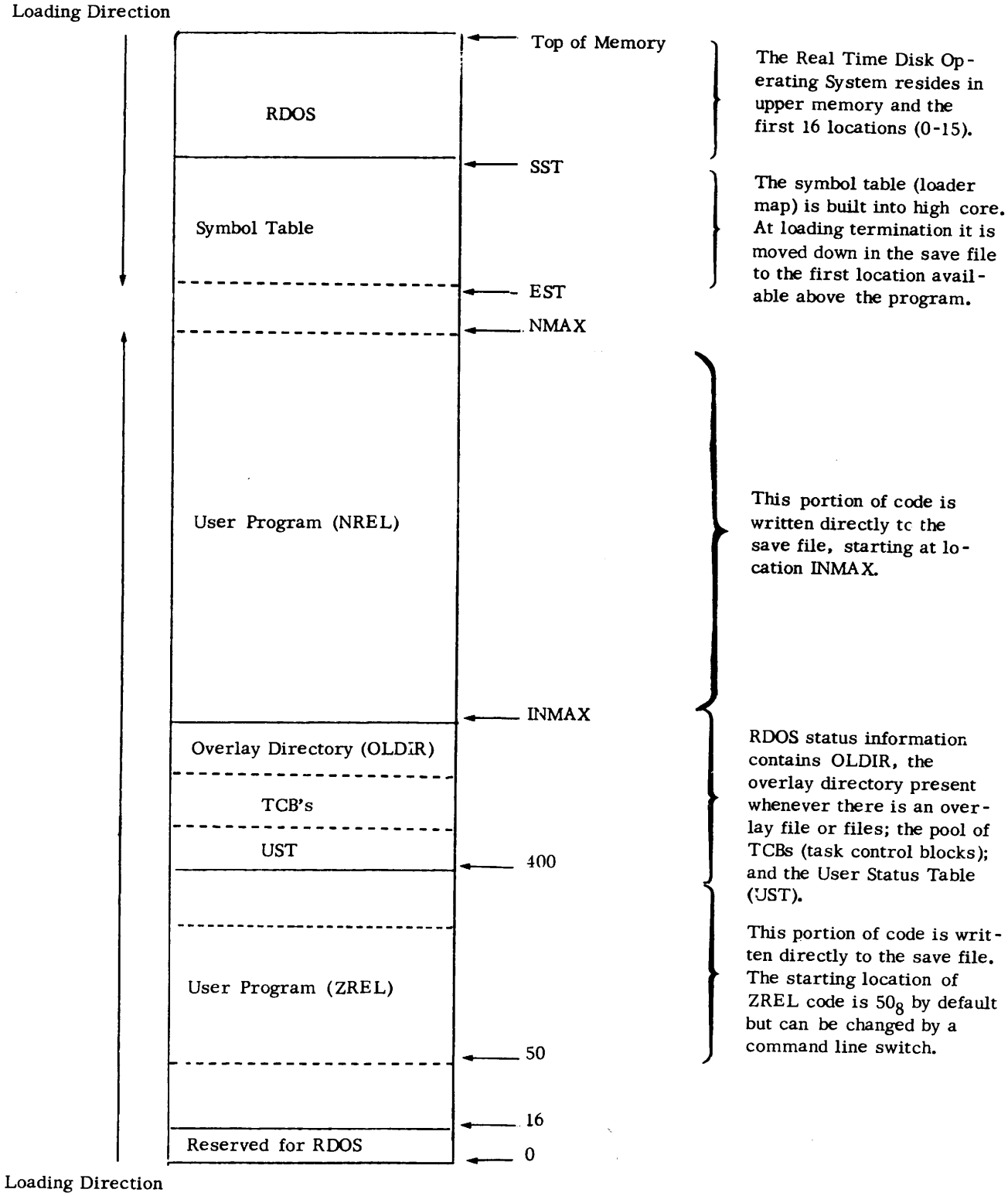
USTBR is the break address (CTRL C). At the termination of loading, the address is set to a -1. If unchanged at run time, whenever a CTRL C break occurs the core image will be written to the file BREAK.SV on the default directory device and return made to the CLI. Alternatively, the user program can set USTBR to an address to which control will be transferred if a CTRL C break occurs. As with CTRL A, the ACs are lost.

USTCH contains the number of program tasks in its left byte, and the number of I/O channels in the right byte.

USTSV is the address of the FORTRAN State Variable Save routine (initialized to 0), which is part of the FORTRAN library. This routine is required in a multitask FORTRAN environment to keep track of each task's FORTRAN run time stack variables. Among the variables handled by this routine are QSP, SP, NSP, and AFSE. For more information about the FORTRAN library, see the FORTRAN Run Time Library User's Manual.

USTSA points to the appropriate task scheduler, except when global switch /C is used in the RLDR command; /C causes USTSA to point to the starting address of the program. When USTSA points to the task scheduler, the scheduler obtains the program starting address from the first task control block. (word TCBPC).

LOADING RELOCATABLE BINARY FILES



LOADING RELOCATABLE BINARY FILES

SST is the start of the symbol table which is the first address below RDOS during loading. EST is the end of the symbol table, which is the first address available below the symbol table during loading. NMAX is the first available address for further loading; and INMAX is the beginning of user NREL code.

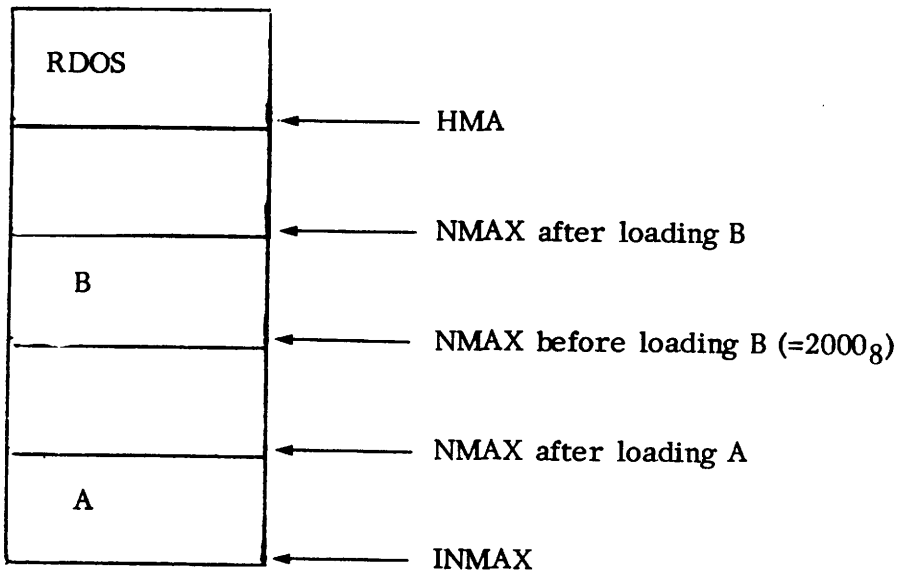
The RDOS relocatable loader permits loading beginning at location 16. ZREL code begins at location 50. Locations 16 - 47 can be reserved by the .LOC pseudo-op at assembly time. INMAX is determined by the number of tasks and overlays. By default, the number of tasks is one.

User Adjustment of NMAX

When loading a number of programs, the user can adjust the value of NMAX. The loader will accept any value of NMAX that is not less than its current value. The value can be adjusted by a local option as shown below:

```
RLDR A 2000/N B )
```

where: 2000/N is a local option giving an adjusted NMAX (2000 octal) at which to begin loading the next program, B.



END OF CHAPTER

APPENDIX A

RDOS OVERLAY LOADER

Supplied as part of the RDOS system is OVLDR.SV, the overlay loader, which may be invoked by the CLI command OVLDR. The overlay loader can be used to create an overlay replacement file. The overlay replacement file can later replace one or more overlays in an existing overlay file. Up to 127 overlays can be replaced.

The format of the OVLDR command is:

```
OVLDR filename overlay-descriptor1/N overlay-list { overlay-descriptor2/N ↑ }
      overlay-list ... } [filename/L] { filename/E }
```

where:

filename is the name of the save file associated with the overlay file in which overlays are to be replaced. The replacement overlay file is named filename.OR.

overlay-descriptor is either a 1 to 6 digit octal number giving the node number/overlay number that identifies the overlay or is the symbolic name of the overlay. The overlay descriptor must be followed by the local /N switch. If the symbolic name is used, it must have been declared in a .ENTO pseudo-op in the save file.

overlay-list is a list of one or more relocatable binaries that are to replace the preceding overlay.

filenames followed by /E and /L are optional error and listing files respectively.

The global switches are:

/A listing of an alphabetical/numeric core map. Note that a local /L switch must also be given to define the listing file for the core map.

/E do not suppress output of error messages to the console. (Used only when there is a listing file (/L) that suppresses console error output.)

/H print all numeric output in hexadecimal. By default output is printed in octal.

The local switches are:

/E preceding file is designated to receive error and information messages.

/L preceding file is designated to receive the core map listing. The map will be numeric unless the global /A switch is given.

/N must follow an overlay-descriptor.

To use OVLDR, the following conditions must exist:

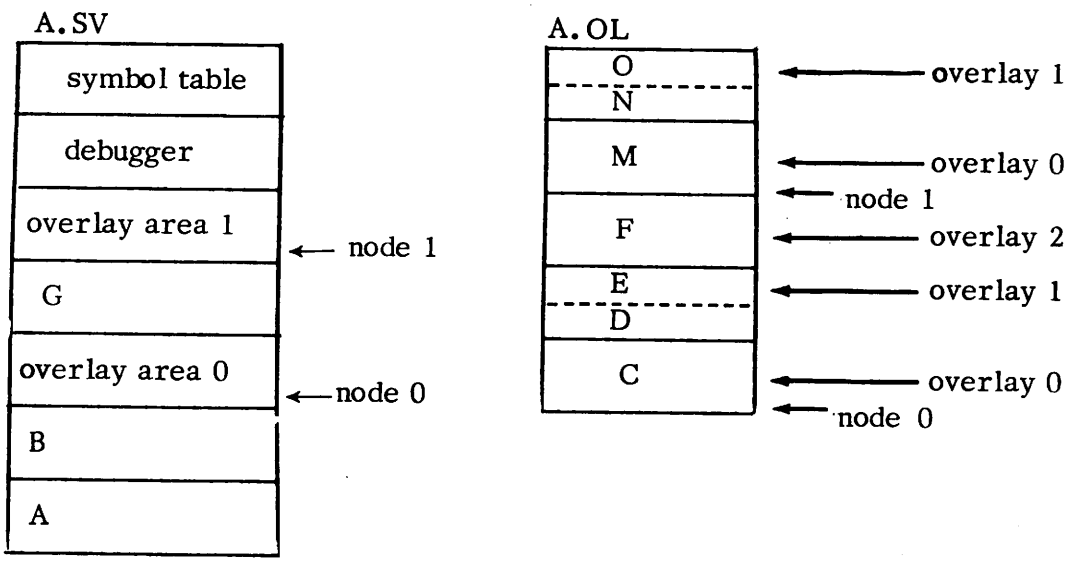
1. There must be an overlay file, filename.OL, created by the RLDR command.

2. The save file, filename.SV, must contain a symbol table.

For example, presume that the following RLDR command was executed:

```
RLDR/D A B [C, D E, F] G [M, N O ])
```

The following save and overlay files were created:



Use of the /D global switch assured loading of the symbol table but also caused the debugger to be loaded. To save core space, the user may wish to load only the symbol table by including it as an external normal in the save file:

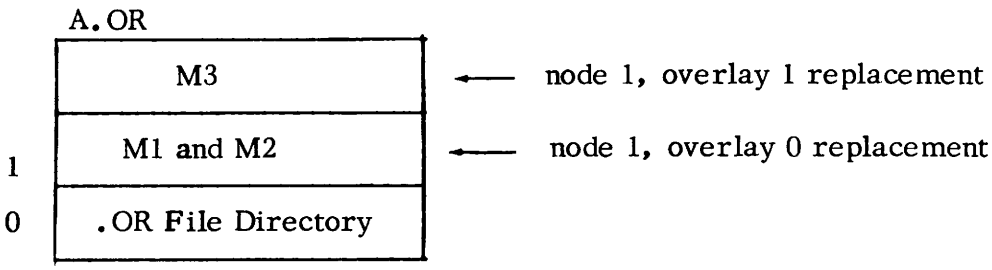
```
.EXTN  .SYM
```

OVLDR can create a file of replacement overlays. For example, to create replacement overlays for M (node 1, overlay 0) and for N-O (node 1, overlay 1), the following command might be given:

```
OVLDR/A/E  A  400/N M1 M2 401/N  M3 OLIST/L )
```

In the example, the error and information listing will be to the console and an alphanumeric core map will be listed in OLIST.

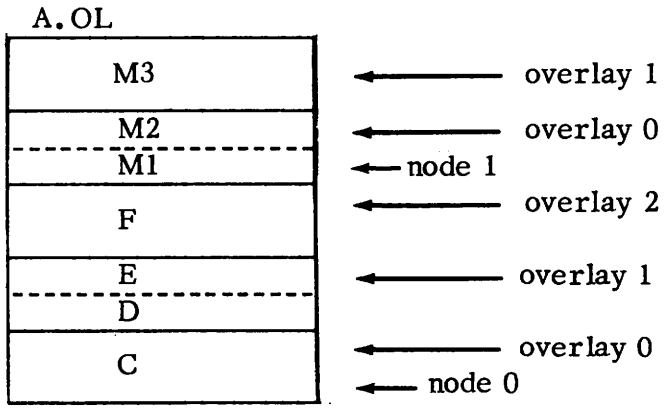
When OVLDR is executed, the file A.OR is created containing the following:



The CLI command, REPLACE, is used to replace current contents of an overlay file with the new overlays created by the OVLDR command. For example, to replace A.OL overlays with the contents of A.OR, the following command would be given:

```
REPLACE  A )
```

When the command to replace is executed, A.OL would appear as follows:



The error messages that may occur in executing an OVLDR command are given below. They are all fatal.

'LLEGAL LOAD ADDRESS

Any attempt to load into an address outside the overlay area.

NO SYMBOL TABLE

The symbol table was not created when the save and overlay files were loaded.

NO OVERLAY DIRECTORY

The save file does not contain the overlay directory, OLDIR.

INSUFFICIENT MEMORY

OVLDR cannot execute in available memory.

COMMON SIZE ERROR

An overlay defines blank COMMON to be larger than that in the save file.

EXTERNAL LOCATION UNDEFINED OR NOT WITHIN OVERLAY

Either a symbol is not defined within the save file or the symbol value is not legal for the overlay area.

END OF APPENDIX

APPENDIX B

RLDR
Symbol Table Formats

Disk Symbol Format

Word 0 Type in left byte, Length of name in words in right

Word 1 Symbol equivalence

Word 2 Node/Overlay word, node/ overlay of definition of the symbol (ENT's only)
Named common size--(COMM's)

Word 3--Word n symbol name packed
2 characters per word.
1st in left, 2nd in right, etc.

The equivalence of an ENTO is the same as the contents of its node overlay word.

All symbols, defined and undefined have a disk entry. Undefined symbols also have entry in core.

Disk symbols are hashed to 20_8 buckets. The algorithm is (1st character, 3rd character, last character) masked to $0\text{-}17_8$. Entries are placed sequentially within a bucket. They are not sorted within the bucket. As many symbols as will fit are placed in each block. This number varies with the size of the entries. The last word in a block is the overlay pointer. It carries the block number of the next block of this bucket. If it is zero, there is not subsequent block.

Titles and local symbols are placed (if loaded) in block 20 and subsequent overflow blocks. All local symbols in a module precede the corresponding title.

Core Symbol Format

Undefined symbols are represented in core as well as on disk.

Word 0: block # and channel bits in high order bits.

Word 1: Offset in left byte. Type in right.
Offset is offset of symbol in disk symbol block.
Type designations are as shown in RLDR manual.
Bit 13 is flag for extended format.

Word 2-n chain pointers.

END OF APPENDIX

DataGeneral

SOFTWARE DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

FOLD DOWN FIRST FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary If Mailed In The United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Software Documentation

FOLD UP SECOND FOLD UP

STAPLE