REAL-TIME SOFTWARE

A REFERENCE TEXT FOR PROGRAMMERS

HEWLETT hp PACKARD

11000 Wolfe Road

Cupertino, California 95014

HP 02116-9139

September 1970

First edition, Feb. 1969
Revised, Sept. 1969
Revised, April 1970
Revised, Sept. 1970

Fourth Edition

PREFACE

REAL-TIME SOFTWARE is the programmer's guide to the Hewlett-Packard Real-Time Executive System-- a real-time, priority-oriented, multiprogramming system using an HP 2116B computer and disc storage. The reader should be familiar with one of the Hewlett-Packard programming languages, as presented in the FORTRAN (02116-9015), ALGOL (02116-9072), and ASSEMBLER (02116-9014) programmer's reference manuals.

This edition supersedes the April, 1970 edition of REAL-TIME SOFTWARE, incorporates all manual changes issued to date, and describes the use of ALGOL with the real-time system.

The introduction explains the software and hardware elements of the system. Section I presents the foundational ideas of the Real-Time Executive, while Section II deals with the actual structure and operation of the executive. Sections III and IV detail the complete set of operator requests and program calls to the executive. All facets of real-time programming -- Editor, FORTRAN, ALGOL, Assembler, Loader, DEBUG, and Library -- are covered in Section V. Section VI assembles all the necessary information on real-time input/output, including priviledged interrupt processing and a sample I/O driver. Procedures for installing and initiating the software appear in Section VII. The appendices provide tables, summaries, samples, and a complete listing of error messages.

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

INTRODUCTION

The Hewlett-Packard Real-Time Executive System incorporates Real-Time FORTRAN, Real-Time ALGOL and Real-Time Assembly Language programs into a sophisticated scheduling and multiprogramming environment. The Real-Time Executive uses multiprogramming and priorities to schedule both real-time and background programs that are core- or disc-resident. All input/output and interrupt processing is controlled by RTE (Real-Time Executive), except for special privileged interrupts which circumvent RTE for quicker response.

The RTE system has two program areas of core for multiprogramming: a real-time program area and a background program area. Each has a disc-resident overlay section for programs to be read in from the disc, and a core-resident program section for high-priority, quick response programs. See Figure 1-1 for the complete layout of core.

Scheduling of all programs is based on priority: if a high priority real-time program has to suspend execution, a lower-priority background program can execute in the interim. Real-time disc-resident programs, in addition, can be suspended and swapped out of core if a higher priority program needs the area. Programs may be scheduled for execution by an operator request, a program request, a device interrupt, or the completion of a time interval.

When a program requests an I/O transfer, RTE places the program in an I/O suspend state, initiates the I/O operation, and starts executing the next highest priority scheduled program. When the I/O transfer is complete, RTE reschedules the suspended program for execution.

Software subsystems that run under control of RTE as background programs provide additional power and convenience. The FORTRAN Compiler, the ALGOL Compiler and the Assembler translate source programs and generate object code directly onto the disc. The Editor creates and updates source statement files on the disc; these files can later be compiled or assembled. In addition to loading programs into the background area for debugging and execution, the Relocating Loader can make permanent changes to the set of disc-resident programs.

FEATURES OF THE REAL-TIME EXECUTIVE SYSTEM

The RTE contains the following highlights and features:

- Real-Time I/O processing
- Privileged interrupts for quick response
- Multiprogramming of real-time and background tasks
- Priority-oriented scheduling of programs, based on clock time or interrupts
- On-line changes to the set of disc-resident programs
- Load-and-Go facility
- Disc storage of programs, source files and data
- Centralized and device-independent I/O
- Modular structure: customized configuration
- Background programming aids: FORTRAN, ALGOL, Assembler, Relocating Loader, Library, Debug, and Editor.

HARDWARE CONFIGURATION

The minimum hardware requirements for an RTE system are:

- An HP 2116B Computer with 16, 24, or 32 K Memory, and:
  - 12578A Direct Memory Access
  - 12579A Extended Arithmetic Unit
  - 12591A Memory Protect
- Fixed Head Disc or Drum Storage Unit
- Time Base Generator
- Operator Console (ASR-33 or ASR-35 teleprinter)

Background translation and execution require additional input, list, and output power, which an extra ASR-35 can supply.

In addition the following optional hardware may be configured in the system to meet the user's needs:

INTRODUCTION

Optional Hardware

- Line Printer (HP 2778A, 120 columns; HP 2778A-01, 132 columns)
- Punched Tape Reader
- High Speed Tape Punch

REAL-TIME SOFTWARE MODULES

The following software tapes are supplied to the user:

- Real-Time Exec Modules
- Real-Time Assembler
- Real-Time FORTRAN Compiler
- Real-Time Relocating Loader
- Real-Time Relocatable Library
- Real-Time I/O Drivers
- RTGEN, the Real-Time Generator
- Real-Time ALGOL Compiler
- Real-Time Editor

Executive Modules

These three program modules, plus I/O drivers and various tables, make up the "Executive" part of a Real-Time Executive System.

- EXEC
- SCHED
- IOC

The Real-Time Executive is described in Section II.

Background Programs

Real-Time FORTRAN, Real-Time ALGOL and Real-Time Assembler consist of a main program and several segments.

The Real-Time Relocatable Library consists of math, service, and I/O subroutines which may be appended to a user program by the Real-Time Relocating Loader.

Real-Time I/O Drivers

The following I/O Drivers are discussed in Section VI:

Name	I/O Device
DVR00	Teleprinter
DVR01	Punched Tape Reader
DVR02	High-Speed Punch
DVR12	Line Printer
DVR30	Disc Memory

RTGEN, the Real-Time Generator

RTGEN is an independent, absolute program that configures complete real-time systems from the Real-Time Software modules, user programs, and information describing the hardware environment.

SECTION I
FUNDAMENTAL CONCEPTS

The Hewlett-Packard Real-Time Executive (RTE) System provides a disc-based environment for priority-scheduled multiprogramming, centralized input/output, and general-purpose background operations. Section I introduces the basic concepts of the RTE System.

Real-time systems collect data, monitor instruments, direct servo-mechanisms, and control laboratory experiments. For example, in an oil refinery, the computer measures temperatures and pressures using a multitude of sensors in liquids and atmospheres, and transducers connected to pipelines and containers. Real-time programs accept the data (which has been converted to binary form), produce status reports, adjust physical factors in the external process, and notify operators in special cases.

MULTIPROGRAMMING

Since real-time systems usually contain several programs to control different conditions or processes, several external events may require the attention of different programs simultaneously. Multiprogramming, a technique in which an executive routine allocates the resources of the computer (such as core storage, I/O channels, disc storage, and execution time) to more than one competing program, allows a single computer system to execute a multiplicity of real-time tasks concurrently.

In some real-time systems, each program must run to completion before the next program executes. Multiprogramming increases the efficiency, response speed, and flexibility of the computer becauses logical delays in one program (e.g., for input) are used to execute other programs. As a result, a single processor interleaves the execution of two or more programs.

Scheduling

In multiprogramming, an external device, an operator, or another program turns on, or schedules, the programs for execution. The executive routine has a scheduling module which decides when to execute the competing programs.

The user of a Hewlett-Packard RTE System defines up to four program areas for his system and decides which programs to run in each area. Programs may be in core at all times (for fast response) or may be disc-resident, in which case they are read into an overlay area of core to execute. Figure 1-1 shows the core layout of a typical Hewlett-Packard RTE System.

Background Operations

A background area is shown in Figure 1-1. Less-critical, general-purpose programs (such as compilers, assemblers, editors, etc.) execute in this background area. While the real-time area is handling external events, these background programs may create new programs to control future processes.

The HP background programs include a FORTRAN Compiler, ALGOL Compiler, Assembler, Relocating Loader, and Editor, and two additional features: creation of source statement files on the disc for ease of editing and translation; and load-and-go, the ability to translate and execute new programs without any paper tape output, and with little operator intervention. In addition, new programs can be added to the set of permanent user programs, and old ones deleted.

PRIORITY

Whenever programs conflict because of simultaneous demands for execution, priorities are necessary. In the Hewlett-Packard RTE System, each program has a priority level assigned by the user. When a conflict occurs, the Real-Time Executive decides in favor of the higher priority program.

The RTE System handles priorities on a completely generalized basis. The highest priority program requiring execution, executes first. Then, if that program suspends execution (possibly for an input wait), the next highest priority scheduled program executes. This process continues down the priority list until a higher priority program is rescheduled.

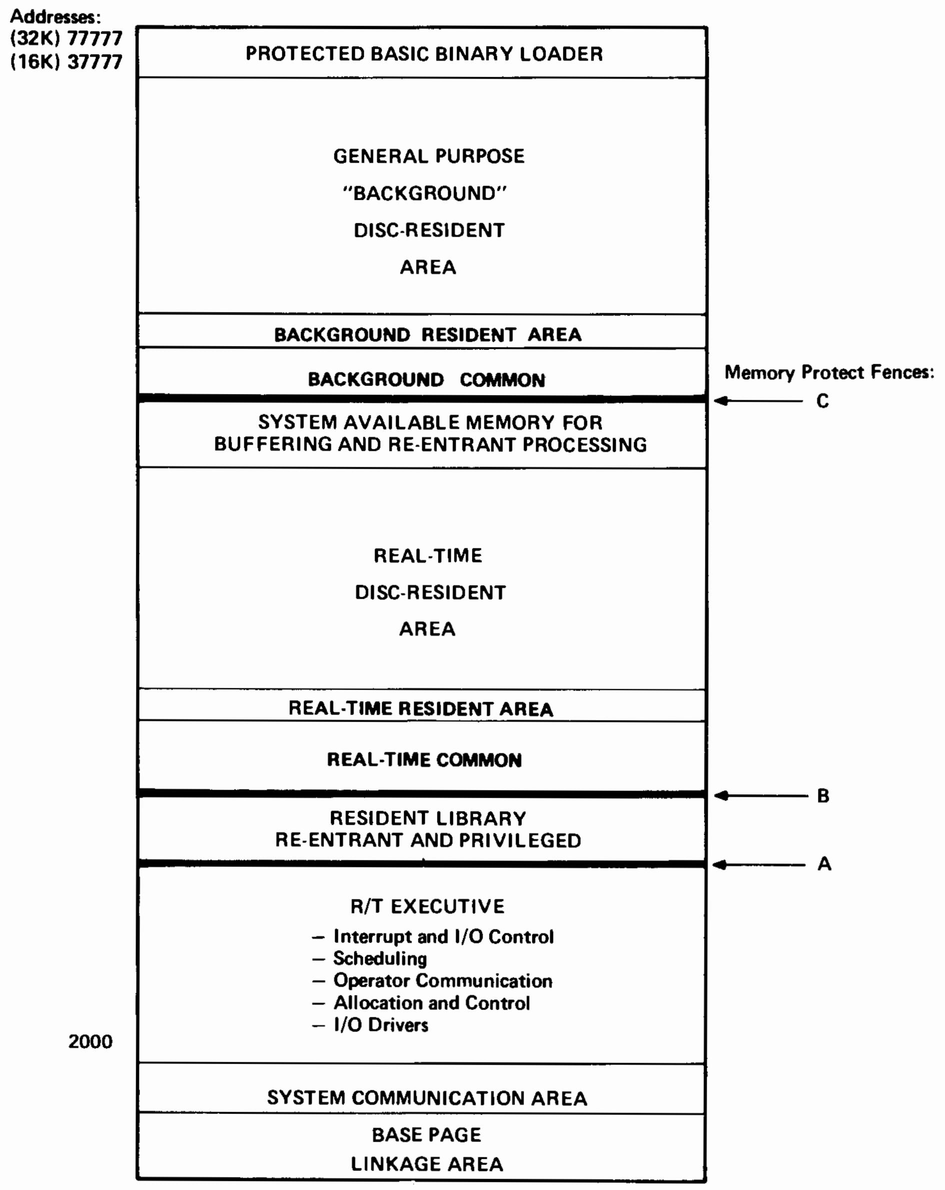
Addresses:
(32K) 77777
(16K) 37777
PROTECTED BASIC BINARY LOADER
GENERAL PURPOSE "BACKGROUND" DISC-RESIDENT AREA
BACKGROUND RESIDENT AREA
BACKGROUND COMMON
Memory Protect Fences:
C
SYSTEM AVAILABLE MEMORY FOR BUFFERING AND RE-ENTRANT PROCESSING
REAL-TIME DISC-RESIDENT AREA
REAL-TIME RESIDENT AREA
REAL-TIME COMMON
B
RESIDENT LIBRARY RE-ENTRANT AND PRIVILEGED
A
R/T EXECUTIVE
– Interrupt and I/O Control
– Scheduling
– Operator Communication
– Allocation and Control
– I/O Drivers
2000
SYSTEM COMMUNICATION AREA
BASE PAGE LINKAGE AREA

Figure 1-1. Core Allocations in the Hewlett-Packard RTE System

NOTE: FORTRAN requires 4K of core. Assembler requires 4K of core. RTE ALGOL requires 8K of core.

SECTION II
REAL-TIME EXECUTIVE

The RTE System divides core into two general parts: the executive area, and the user program area. The executive part is resident in the lower segment of core and has several sub-areas: the RTE program modules, the I/O drivers (see Section VI), the system table area, and the base page area. The user program area is located above the executive and may have several parts as described below.

A memory protect boundary is set between the executive area and a user program area. Whenever a user program attempts to execute an I/O instruction (including a HALT) or to modify the executive area, a memory protect interrupt is generated. This interrupt causes the executive to gain control and the user program will be abnormally terminated. Programs that run in the user area must use EXEC calls (Section IV) for input/output, termination, and suspension.

USER PROGRAM AREAS

A particular RTE System may include up to four different user program areas, corresponding to four program types. Each program, whether core or disc-resident, has an associated ID segment in the resident executive area that records the static and dynamic information defining the program. (See Appendix A for a full treatment of the ID segment.)

The program areas are created at system generation time (see Section VII), and if a system does not include any programs of one type, that area is left out of the system. (A real-time disc-resident area may be specified for the addition of programs at a later date.) The four types (whose areas are diagrammed in Figure 1-1) are described below.

Real-Time Core-Resident Programs

This type is always resident along with RTE for quick resonse to high priority needs. Programs of this type may control a logically subordinate set of disc-resident programs. Core-resident programs should not use FORTRAN I/O operations since the utility formatting routine is quite large, nor use any large utility library routines. (See Section V.)

Real-Time Disc-Resident Programs

This type resides in absolute format on the disc and must be transferred into a reserved part of core before executing. Thus, disc-resident programs provide slower response than core-resident. Since all real-time disc-resident programs have the same point of origin in core, only one program may be in core at a time. With the optional swapping feature, programs of lower priority or suspended programs (except I/O suspended) are swapped out of this region, and waiting programs are swapped in.

Background Core-Resident Programs

These programs are identical to the real-time core-resident type, except that the program area is located at the start of the background region and shares a common area with the background disc-resident programs.

Background Disc-Resident Programs

Unlike the real-time disc-resident type, background disc-resident programs are never swapped, but may be segmented into a main program and several overlay segments. These disc-resident programs have the same point of origin in core, and only one resides in memory at a time. The FORTRAN Compiler, ALGOL Compiler, Assembler, Editor, and Relocating Loader are background disc-resident programs.

User programs written for the background region may be segmented; the procedure is discribed in Section V, REAL-TIME PROGRAMMING.

The set of disc-resident programs (both real-time and background) may be modified on-line using special features of the Relocating Loader. (See Section V.)

Subroutines

Each user program (main or segment) may consist of a primary routine, containing the transfer point for entry into the program from RTE, and a series of subroutines. In assembly language, the transfer point is the location of the label appearing in the END statement. In FORTRAN, the transfer point is the first executable instruction in a routine containing a PROGRAM statement. RTGEN links the primary routine with its subroutines (which are defined by external references within the primary routine) when the system is generated.

The Relocatable Library consists of subroutines which may be linked to user programs. (See Section V.) Each subroutine is either re-entrant, privileged, or utility. Re-entrant and privileged subroutines may be used by more than one program at a time; if they are referenced by resident programs, RTGEN places them in the resident library. (See Figure 1-1.) If called by disc-resident programs but not in the resident library, the re-entrant and privileged subroutines are appended to the absolute version of the calling program.

Subroutines which cannot be shared because of internal design or I/O considerations are utility subroutines, and a copy of the utility subroutine is appended to each primary routine, whether core or disc-resident, that calls it. RTGEN stores all library programs that are not included in the resident library on the disc in relocatable format (as utility routines to be used by the Relocating Loader).

RTE AREA

The Real-Time Executive modules handle program scheduling, input/output operations, interrupt processing, and user communication as described on the following pages.

Scheduling Module (SCHED)

The Scheduling Module (SCHED) of RTE schedules programs for execution, monitors the time list (to schedule programs at specific times), transfers disc-resident programs into core, swaps real-time disc-resident programs, and processes operator requests.

Scheduling, the dynamic allocation of RTE System resources in a real-time environment, is based on program state and program priority. Programs may be in one of four states:

- Executing,
- Scheduled,
- Suspended, or
- Dormant.

The status field in the ID segment (see Appendix A) records the state of the program.

The scheduled program with the highest priority is in the executing state and has first access to execution time until one of the following conditions occurs:

- The program terminates,
- A higher priority program is scheduled, or
- The program suspends.

Programs of a lower or equal priority that are waiting to execute are in the scheduled state. These programs may be in core or on the disc.

Programs that were interrupted in the executing state to wait for an event to occur before re-scheduling are in the suspended state. Programs may be suspended for several reasons:

- Waiting for the completion of an I/O operation,
- Waiting for the availability of needed memory space,

- Waiting for the completion of a disc allocation,
- Waiting for the completion of a program scheduled by the suspended program,
- The operator has requested that a program be suspended, or
- The program has requested that it be suspended.

Programs which are not currently in either the scheduled, executing, or suspended state are in the dormant state.

Scheduling of a program for execution occurs because of:

- Operator request. (See Section III.)
- Program Request. (See Section IV.)
- External interrupt.
- Completion of a time interval (the ID segment contains the time specifications for the program).

PRIORITY LEVEL

Program priority determines the order of a program in the scheduled, suspended, and dormant states. The priority field of the ID segment (see Appendix A) records the priority of the program. Priorities range from 0 (the highest, reserved for system programs) to 99 (the lowest). The priority of any dormant program may be changed by an operator request, and more than one program may be at the same priority.

For each program state except executing, RTE maintains an ordered list of the programs in that state, connecting the ID segments according to the priority of the programs. There are three types of lists:

- Scheduled
- Suspended
- Dormant

The Base Page Communication Area (see Appendix A) contains the pointers to the ID segment of the first, or highest priority, program in each list. Then, the linkage field of each ID segment contains the location of the next ID segment in the list. There are one scheduled list, one dormant list, and four types of suspension lists:

- I/O suspension lists (one for each device),
- Memory availability list,
- Disc allocation list,
- Operator suspension list.

OPERATOR REQUESTS

Although the RTE System requires little operator intervention during normal functioning, the operator retains ultimate control of the system with a series of 16 requests entered through the teleprinter keyboard. (See Section III.) Operator requests, which are handled by SCHED, can interrupt RTE at any time to:

- Turn programs on and off.
- Suspend and restart programs.
- Examine the status of any program or I/O device.
- Schedule programs to execute at specified times.
- Change the priority of dormant programs.
- Set up load-and-go operations and source files.
- Declare I/O devices up or down.
- Dynamically alter the logical I/O structure and buffering designations.
- Eliminate disc-resident programs from the system.
- Initialize the real-time clock and print the time.

Input/Output Control Module (IOC)

IOC is responsible for processing all system interrupts and input/output operations. Section VI describes the I/O structure of the RTE System in detail, especially I/O drivers, and privileged interrupt processing.

INTERRUPT PROCESSING

All interrupts, except privileged interrupts, cause a transfer to a central routine in IOC, the Central Interrupt Control, which is responsible for saving and restoring the various registers, analyzing the source of the interrupt, and calling the appropriate processing routine.

An interrupt table, ordered by hardware interrupt priority, records the correct processor routine for each interrupt. (See Section VI.) Processors that respond to standard system interrupts (real-time clock routine, memory protect, standard I/O drivers) are called directly by CIC. Processors that respond to user-controlled devices or interrupt sources are scheduled just like other programs.

When an interrupt occurs, the instruction in the word corresponding to the I/O channel number is executed. For all active interrupt locations, except privileged interrupts, this instruction is a jump subroutine (indirect) to CIC. The special processing for privileged interrupts is described in Section VI.

INPUT/OUTPUT PROCESSING

IOC allocates DMA channels for I/O devices, provides for referencing I/O devices by logical unit number rather than directly by EQT entry number or I/O channel, stacks program I/O requests by priority of the calling program, and provides automatic output buffering, when specified, for low- to medium-speed devices.

I/O drivers are under control of IOC and CIC for initiation and completion of program-requested I/O operations; they provide simultaneous multi-device control.

Program requests for I/O are made by EXEC calls which specify the type of transfer and device desired. IOC handles the request, suspending the requesting program until the operation is complete. All input/output operations occur concurrently with program execution. However, the transfer is considered non-buffered, since the requesting program is suspended and the next lower priority scheduled program is allocated execution time during the suspension.

Exec Communication Module (EXEC)

EXEC processes program EXEC calls. (See Section IV.) When an executing program makes an EXEC call, it attempts to execute a jump subroutine to EXEC in the protected area of core. This causes a memory protect violation interrupt, which CIC recognizes, and transfers to EXEC. EXEC examines the parameters associated with the jump subroutine in the calling program. If the parameters are legal, EXEC handles the request itself or calls an appropriate part of RTE to process it.

Using EXEC calls, which are the line of communication between an executing program and RTE, a program is able to:

- Perform input and output operations,
- Allocate and release disc space,
- Terminate or suspend itself,
- Load its segments (if background disc type),
- Schedule other program and set execution time cycles,
- Obtain the time of day.

SECTION III
OPERATOR REQUESTS

The operator retains ultimate control over an executing Real-Time Executive System by means of 16 powerful operator requests. From the teleprinter console, he can interrupt RTE at any time to:

- Turn programs on and off.
- Suspend and restart programs.
- Examine the status of any program or I/O device.
- Schedule programs to execute at specified times.
- Change the priority of dormant programs.
- Set up load-and-go operations and source files.
- Declare I/O devices up or down.
- Dynamically alter the I/O structure and buffering.
- Eliminate disc-resident programs from the system.
- Initialize the real-time clock and print the time.

The operator gains the attention of RTE by pressing any key on the console. When the computer responds with an asterisk (*), the operator inputs any operator request, consisting of a two-character request word (e.g., ON, UP, etc.) and up to seven parameter fields separated by commas (two commas in a row mean a parameter is zero). The second field is often a program name. The request formats include items in italics and items in brackets. The italicized items are symbolic, and the bracketed items are optional. The "control-A" character acts to eliminate the previous character; a "rubout" anywhere in a request eliminates the request. The "control" and "A" keys struck simultaneously, produce the "control-A".

Each operator request is described in detail in this section; but for a quick reference, refer to the summary in Appendix D.

ON

Purpose

To schedule a program for execution. Up to five parameters may be passed to the program.

Format

ON, *name* [,NOW] [, *P1, P2,..,P5*]

where *name* is the name of a program,

NOW schedules a program immediately that is normally scheduled by the clock time (see IT), and

P1 through *P5* is a list of parameters passed to *name* when it is scheduled (must be positive integers less than 32767).

Comments

The ON operator request schedules the various background programming aids (i.e., *name* equals EDIT, FTN, ALGOL, ASMB, or LOADR). ON passes parameters to these programs. (See Section V for details.)

If the resolution code in the NAM record of the program is not zero, RTE places the program in the time list for execution at specified times (unless NOW appears, in which case, the program executes first, then goes into the time list).

OF

Purpose

To terminate a program, or to remove a background program which was loaded on-line but not permanently incorporated into the protected RTE system.

Format

OF, *name*, *P*

where *P* is an integer, and
name is the name of a program.

P = Ø	terminates and removes from the time list any executing, scheduled, or operator suspended program; terminates programs which are I/O, memory, or disc suspended the next time they are scheduled. In neither case are disc tracks released.
P>Ø,≠8	terminates immediately the program named, removes it from time list, and releases all disc tracks. If suspended for I/O, the device and channel are cleared by a CLC.
P=8	same as for P>Ø, plus program is completely removed from the core RTE system. Should be used only on programs loaded on-line, but not permanently incorporated into the system. The ID segment is blanked, and the tracks containing the program (if not permanent) are released.

Comments

When *P* = 8, the program's ID segment becomes available for loading another program with LOADR. For programs with segments, OF must be used on the segments as well as the main program. OF is not used to remove permanent programs because their ID segments on the disc are not altered by this request.

If *p* = 8 and *name* is I/O suspended, the OF request is treated as if *p* were greater than Ø, but not equal to 8. OF,*name*,8 must be entered again to permanently remove *name* from the RTE System.

SS

Purpose

To suspend a program from execution.

Format

SS, *name*

where *name* is the name of the program to be suspended.

Comments

The SS request places the program in the operator suspended list immediately if the program is executing, scheduled, or already there. If the program is dormant, the request is illegal. If the program is suspended already for I/O, memory, or disc, RTE waits until the current suspend is over, then suspends the program again.

The SS operator request is similar to the PROGRAM SUSPEND EXEC CALL. (See Section IV.)

GO

Purpose

To reschedule a program that has been suspended by an SS operator request or a SUSPEND EXEC call. (See Section IV.)

Format

GO, *name* [, *P1*, ... , *P5*]

where *name* is the name of an operator suspended program to be scheduled for execution

P1 through *P5* is a list of parameters to be passed to *name*

Comments

If the program has not been suspended previously by the operator or has not suspended itself, the request is illegal.

When a program resumes execution after suspending itself, the address of the parameters, if any, passed by GO, is in the B register. In FORTRAN, an immediate call to the library subroutine RMPAR retrieves the parameters. (See Section IV, SUSPEND EXEC CALL.)

ST

Purpose

To request that the status (priority, current list, time values) of a program be printed on the console.

Format

ST, *name*

where *name* is the name of the program whose status is to be printed.

Comments

The status is printed on one line in a fixed format:

PR S R MPT HR MI SC T

where *PR* is the priority, a value from 1 to 99_{10},

S is the current list in which the program is located:

Ø - Dormant

1 - Scheduled

2 - I/O suspend

3 - Not used

4 - Unavailable memory suspend

5 - Disc allocation suspend

6 - Operator suspend

R, MPT, HR, MI, and *SC* are all zero (Ø), unless the program is scheduled by the clock (see IT, this section, for the meaning of these items).

The letter 'T' appears when the program is currently in the time list (as the result of an ON operator request).

IT

Purpose

To set time values for a program, so that the program executes automatically at selected times when turned ON.

Format

IT, *name*, *R*,*MPT* [, *HR*, *MI* [,*SC* [,*MS*]]]

where *name* is the name of the program,

R is the resolution code:

1 - Tens of milliseconds

2 - Seconds

3 - Minutes

4 - Hours

MPT is a number from Ø to 999 which is used with *R* to give the actual time interval for scheduling (see Comments),

HR - Hours
MI - Minutes
SC - Seconds
MS - Tens of ms.

} Sets an initial start time.

Comments

The time values for each program are originally set when the program is incorporated into the system by RTGEN or LOADR. Whenever the RTE system starts up from the disc, these values are in effect. RTE stores the time values set by IT in core, but not on the disc since it is protected from alteration. Therefore, when the system restarts from the disc, any values set by IT are lost, and the original time values are reinstated.

The resolution code (*R*) gives the units for the multiple execution interval (*MPT*). Thus, if *R*=2 and *MPT*=1ØØ then *name* would be scheduled every 1ØØ seconds. If *HR, MI, SC,* and *MS* are present, the first execution occurs at the initial start time which these parameters specify.

The IT operator request is similar to the EXECUTION TIME EXEC CALL. (See Section IV).

PR

Purpose

To change the priority of a program.

Format

PR, *name, n*

where *name* is the name of the program,
n is the new priority.

Comments

The priority of *name* resets (to that set by RTGEN or LOADR) whenever the RTE system restarts from disc. One is the highest priority, and 99 is the lowest.

Only *dormant* programs can have their priority changed.

LG

Purpose

To allocate or release a group of disc tracks for load-and-go operations.

Format

LG, *n*

where *n* is:

Ø (zero)	- Release the *allocated* load-and-go area.
n (>Ø)	- Allocate *n* contiguous tracks for a load-and-go area; set flags for load-and-go operation.

Comments

LG must allocate enough tracks for storing binary object code *before* a load-and-go compilation or assembly. If not, the compiler or assembler aborts and a diagnostic appears:

I0Ø6	- Load-and-go area not defined.
I0Ø9	- Overflow of load-and-go area.

For a general description of the load-and-go facility, see Section I, FUNDAMENTAL CONCEPTS. For details, see Section V, REAL-TIME PROGRAMMING.

LS

Purpose

To designate the disc logical unit number and starting track number of an existing source file, subsequent to operating on it with EDIT, FTN, ALGOL, or ASMB.

Format

LS, *P1*, *P2*

where *P1* is the logical unit number of the disc containing the source file.

P1 = 2 or 3 System or auxiliary disc units.

P1 = Ø Eliminate the current source file designation.

P2 is the starting track number of the source file (in decimal).

Comments

LS replaces any previous file declarations with the current file. Only one file may be declared at a time.

For details on creating, updating, compiling, or assembling source files, see Section V, REAL-TIME PROGRAMMING.

DN

Purpose

To declare an I/O device down (i.e., unavailable for use by the RTE system).

Format

```
DN, n
```

where *n* is the Equipment Table (EQT) entry number of the I/O device to be set down. (See Section VI, REAL-TIME INPUT/OUTPUT, for details on the EQT.)

Comments

The device set down is unavailable until set up by the UP operator request. The operator might set a device down because of equipment problems, tape change, etc.

UP

Purpose

To declare an I/O device up (i.e., available for use by the RTE system).

Format

UP, *n*

where *n* is the EQT entry number of the device to be set up. (See Section VI, REAL-TIME INPUT/OUTPUT, for details on EQT.)

Comments

When the operator or the RTE system has set an I/O device down for some reason, the operator should correct the situation before declaring the device available again with the UP operator request. If the problem is irrecoverable, the operator can use LU to switch the logical unit number assignment to another device (see LU, this section).

LU

Purpose

To print or change a logical unit number assignment.

Format

LU, *n*[,*m*]

where *n* is a logical unit number from 1 to 63_{10},
m is an EQT entry number to assign to *n*.
(If *m* = Ø, the current assignment of logical unit *n* is released; if *m* is absent, the assignment of logical unit *n* is printed.)

Comments

Section VI, REAL-TIME INPUT/OUTPUT, explains logical unit numbers and equipment table entry numbers in detail. When the system is restarted from the disc, any assignments made by LU are reset to those originally set by RTGEN.

Logical unit assignments are printed on one line:

LU #*n* = #*x*

where *n* is the logical unit number and *x* is the EQT entry assigned to *n*.

When an irrecoverable problem occurs on an I/O device, the operator can bypass the downed device by reassigning the logical unit number to an operable device on another channel. Any programs referencing the downed device are suspended until the device is declared up (see UP).

EQ

Purpose

To print the description and status of an I/O device, as recorded in EQT entry. (See Section VI for details of the EQT entry.)

Format

EQ, *n*

where *n* is the EQT entry number of the I/O device.

Comments

The information is printed as:

SC DVR*nn* *D* *B* U*n* *LS*

where *SC* is the select code (I/O channel).

DVR*nn* is the driver routine,

D is D if DMA required, Ø if not,

B is B if automatic output buffering used, Ø if not,

U*n* is the unit number for subchannel addressing,

LS is the logical status:

Ø - available

1 - unavailable (down)

2 - unavailable (busy)

3 - waiting for DMA assignment

EQ,N,P

Purpose

To change the automatic output buffering designation for a particular I/O device. (See Section VI for details on automatic output buffering.)

Format

```
EQ, n, p
```

where *n* is the EQT entry number, and
p is Ø to delete buffering, 1 to specify buffering.

Comments

When the system is restarted from the disc, buffering designations made by EQ,*n*,*p* are reset to the values originally made by RTGEN.

The standard line printer (controlled by I/O Driver DVR12) may not be buffered.

TM

Purpose

To initialize the real-time clock.

Format

```
TM, DAY, HR, MI, SC
```

where *DAY* is a three-digit day of the year,

HR, MI, and *SC* is the current time on a 24-hour clock.

Comments

The operator usually gives TM in response to a message printed when the RTE system is initiated from the disc:

```
INIT REAL TIME CLOCK
```

TI

Purpose

To print the current time of day and day of the year, as recorded in the real-time clock.

Format

```
TI
```

Comments

The computer prints out the day and time:

DAY HR MI SC

where *DAY* is the three-digit day of the year, and *HR, MI, SC* is the time on a 24-hour clock.

The TI operator request is similar to the TIME REQUEST EXEC CALL. (See Section IV.)

ERROR MESSAGES

RTE rejects operator requests for various reasons. When a request is in error, RTE prints one of the messages below. The operator enters the request again, correctly.

ERROR MESSAGES ON OPERATOR REQUESTS

Message	Meaning
OP CODE ERROR	Illegal operator request word.
NO SUCH PROG	The *name* given is not a main program in the system.
INPUT ERROR	A parameter is illegal.

SECTION IV
EXEC CALLS

Using EXEC calls, which are the line of communication between an executing program and RTE, a program is able to:

- Perform input and output operations.
- Allocate and release disc space.
- Terminate or suspend itself.
- Load its segments (if background disc).
- Schedule other programs.
- Obtain the time of day.
- Set execution time cycles.

An EXEC call is a block of words, consisting of an executable instruction and a list of parameters defining the request. The execution of the instruction causes a memory protect violation interrupt and transfers control into RTE. RTE then determines the type of request (from the parameter list) and, if it is legally specified, initiates processing of the request. The executable instruction is a jump subroutine (JSB) to EXEC, the primary entry point of RTE.

In FORTRAN, EXEC calls are coded as CALL statements. In ALGOL, EXEC calls must be declared a CODE Procedure and parameters must be declared either as NAME or VALUE. In Assembly Language, EXEC calls are coded as JSB, followed by a series of parameter definitions. For any particular call, the object code generated for the FORTRAN CALL Statement is equivalent to the corresponding Assembly Language object code.

This section describes the basic formats of FORTRAN, ALGOL and Assembly Language EXEC calls, then each EXEC call is presented in detail, and finally, the error messages associated with EXEC calls are listed.

FORMAT OF THE ASSEMBLY LANGUAGE CALLING SEQUENCE

The following is a general model of an EXEC call in Assembly Language:

```
          EXT     EXEC          (Used to link program to RTE)
           .
           .
           .
          JSB     EXEC          (Transfer control to RTE)
          DEF     * + n + 1     (Defines point of return from RTE, n
                                is number of parameter's; may not be
                                an indirect address)
          DEF     P1  ⎫         (Define addresses of parameters which
           .          ⎬         may occur anywhere in program; may be
           .          ⎭         multi-level indirect)
          DEF     Pn
          return point          (Continue execution of program)
           .
           .
           .
           .
           .
           .
      P1  ---  ⎫
       .       ⎪
       .       ⎬                (Actual parameter values)
       .       ⎪
      Pn  ---  ⎭
```

FORMAT OF THE FORTRAN CALLING SEQUENCE

In FORTRAN, the EXEC call consists of a CALL Statement and a series of assignment statements defining the variable parameters of the call:

CALL EXEC (*P1*, *P2* .... , *Pn*)

where *P1* through *Pn* are either values or variables defined elsewhere in the program. Variables must begin with a letter I through N, since they are integer variables.

Example

```
      CALL EXEC (7)        )
          or               |     Equivalent calling sequences
       IRCDE = 7           |
      CALL EXEC (IRCDE)    )
```

Some EXEC call functions are handled automatically by the FORTRAN compiler or special subroutines. (Refer to "FORTRAN," Section VI, REAL-TIME PROGRAMMING, and the specific EXEC calls in this section.)

FORMAT OF THE ALGOL CALLING SEQUENCE

In ALGOL, certain conventions must be followed in making EXEC calls. First, since EXEC is external to the program, it must be declared a CODE procedure. Second, parameters that are going to be changed must be declared "name" and those that are not to change must be VALUE parameters. Third, since ALGOL requires that the format of each procedure call be defined, a program must declare a dummy external procedure for each type of EXEC call it makes. (These dummy procedures must be compiled as separate procedures to provide proper linkage in the Loader.)
For example,

```
       (In the main program):
                 :
                 PROCEDURE EXECA(A); INTEGER A;CODE;
                 PROCEDURE EXECB(A,B,C,D); INTEGER A,B,C,D;CODE;
                 :
                 EXECA(I);
                 :
                 EXECB(J,K,L,M);
                 :
                 END$
       (External)
                 HPAL,P,B,L,"EXECA"
                 PROCEDURE EXECA(A); INTEGER A; BEGIN PROCEDURE EXEC(A); INTEGER A;
                 CODE; EXEC(A);
                 END;
```

READ/WRITE

Purpose

To transfer information to or from an external I/O device or a disc storage unit. The program is placed in the I/O suspend list until the operation is complete.

Assembly Language

```
              EXT     EXEC
               .
               .
               .
              JSB     EXEC          (Transfer control to RTE)
              DEF     *+ 5(or 7)    (Point of return from RTE; 7 is for
                                     disc request)
              DEF     RCODE         (Request code)
              DEF     CONWD         (Control information)
              DEF     BUFFR         (Buffer location)
              DEF     BUFFL         (Buffer length)
              DEF     DTRAK         (Track number-disc transfer only)
              DEF     DSECT         (Sector number-disc  transfer only)
              return point          (Continue execution)
               .
               .
RCODE         DEC     1(or2)        (1=READ, 2=WRITE)
CONWD         OCT     conwd         (conwd is described in Comments)
BUFFR         BSS     n             (Buffer of n words)
BUFFL         DEC     n(or -2n)     (Same n; words(+) or characters(-)
DTRAK         DEC     f             (Track number, decimal)
DSECT         DEC     q             (Sector number, decimal)
```

FORTRAN

I/O transfers to regular devices are programmed by standard READ and WRITE statements in FORTRAN. I/O on the disc is done with a library subroutine BINRY (described in Comments).

Comments

CONTROL WORD

The control word (*conwd*), required in the Assembly Language calling sequence, contains several fields defining the nature of the data transfer:

	Ø	Ø	(not used)				A	K	V	M	LOGICAL UNIT #					
BITS	15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø

where A, if 1, designates punching ASCII characters on the teleprinter (M = Ø). ASCII is usually printed; but since it is sometimes desirable to punch ASCII tapes, this option is provided. (If A = Ø, M determines mode of transfer.)

K, if 1, causes keyboard input to be printed as received. If K = Ø, input from the keyboard is not printed.

V, if 1, and M = 1, causes the length of punched tape input to be determined by the word count in the first non-zero character read in. If V = Ø, and M = 1, the length of the punched tape input is determined by the buffer length specified in the EXEC call.

M, determines the mode of data transfer (if applicable) Ø for ASCII format, 1 for binary.

In an Assembly Language calling sequence, the buffer length can be words (+) or characters (-). While the operation is going on, the program is suspended. When the transfer is complete, RTE reschedules the program.

End-of-operation information is transmitted to the program in the A and B registers. The A register contains word 5 of the device EQT entry; the B register contains a positive number which is the number of words or characters (depending upon which the program specified) actually transmitted.

BINRY

FORTRAN programs must call BINRY, the disc read/write library subroutine, to transfer information to or from the disc. The call must specify a buffer array, the array length in words, the disc logical unit number, track number, sector number, and offset in words within the sector. (If the offset equals Ø, the transfer begins on the sector boundary; if the offset equals *n*, then the transfer skips *n* words into the sector before starting.) BINRY has two entry points: BREAD for read operations and BWRIT for write operations.

For example,

```
      DIMENSION IBUF (1Ø), BUF (2Ø)
      LUN = 2
      ITRK = 12
      ISECT = 63
      IOFF = Ø
       .
       .
       .
             ⎧ BREAD ⎫ ⎧BUF    4Ø ⎫
      CALL   ⎨  or   ⎬(⎨ or,   or,⎬ LUN, ITRK, ISECT, IOFF)
             ⎩ BWRIT ⎭ ⎩IBUF   1Ø ⎭
```

I/O CONTROL

Purpose

To carry out various I/O control operations, such as backspace, write end-of-file, rewind, etc. The program is placed in the I/O suspend list until the control operation is complete.

Assembly Language

```
          EXT   EXEC
           .
           .
           .
          JSB   EXEC            (transfer control to RTE)
          DEF   *+ 4(or 3)      (point of return from RTE)
          DEF   RCODE           (request code)
          DEF   CONWD           (control information)
          DEF   PARAM           (optional parameter)
          return point          (continue execution)
           .
           .
RCODE     DEC    3              (request code = 3)
CONWD     OCT    conwd          (see Comments)
PARAM     DEC    n              (required for some control functions;
                                see Comments)
```

FORTRAN

Use the FORTRAN auxiliary I/O statements or an EXEC call sequence.

```
        IRCDE = 3       (Request code)
        ICNWD = conwd   (See Comments)
        IPRAM = x       (Optional; see Comments)
        CALL EXEC (IRCDE, ICNWD, IPRAM)
        CALL EXEC (IRCDE, ICNWD)
```

Comments

CONWD

The control word value (*conwd*) has two fields:

			FUNCTION CODE (see below)								LOGICAL UNIT NUMBER					
BITS	15	14	13	12	11	1Ø	9	8	7	6	5	4	3	2	1	Ø

Function Code (Octal)	Action
ØØØ	Unused
ØØ1	Write end-of-file (magnetic tape)
ØØ2	Backspace one record (magnetic tape)
ØØ3	Forward space one record (magnetic tape)
ØØ4	Rewind (magnetic tape)
ØØ5	Rewind standby (magnetic tape)
ØØ6	Unused
ØØ7	Set end-of-paper tape
Ø1Ø	Generate paper tape leader
Ø11	List output line spacing

Function Code 11_8, list output line spacing, requires the optional parameter mentioned in the calling sequences. PARAM and IPRAM must designate the number of lines to be spaced on the specified logical unit. A negative parameter specifies a page eject on a line printer. For details of line printer formatting consult Appendix G.

I/O STATUS

Purpose

To request information (status condition and device type) about the device assigned to a logical unit number.

Assembly Language

```
          EXT     EXEC
           :
          JSB     EXEC         (Transfer control to RTE)
          DEF     *+4(or 5)    (Point of return from RTE)
          DEF     RCODE        (Request code)
          DEF     CONWD        (Control information)
          DEF     STAT 1       (Status word 1)
          DEF     STAT 2       (Status word 2 - optional)
          return point
           :
RCODE     DEC     13           (Request code = 13)
CONWD     DEC     n            (Logical unit number)
STAT 1    NOP                  (Word 5 of EQT entry returned here)
STAT 2    NOP                  (Word 4 returned here, optional)
```

FORTRAN

```
       IRCDE = 13      (Request code)
       ICNWD = nnB     (nn is octal logical unit)
       CALL EXEC (IRCDE, ICNWD, ISTA1, ISTA2)
```

Comments

The calling program is not suspended. The second status word is optional in the I/O STATUS EXEC call. For a description of words 4 and 5 of the EQT, see Section VI, REAL-TIME INPUT/OUTPUT.

DISC ALLOCATION

Purpose

To request that RTE assign a specified number of contiguous mass storage tracks to the calling program for data storage. Once assigned, the tracks can be released and modified only by the program that requested them. (See DISC RELEASE EXEC CALL.)

Assembly Language

```
          EXT     EXEC
           :
          JSB     EXEC      (Transfer control to RTE)
          DEF     * + 6     (Point of return from RTE)
          DEF     RCODE     (Request code)
          DEF     ITRAK     (Number of contiguous tracks required)
          DEF     STRAK     (Start track number)
          DEF     DISC      (Logical unit number)
          DEF     ISECT     (Number of sectors/track)
          return point      (Continue execution)
           :
RCODE     DEC     4         (Request code = 4)
ITRAK     DEC     n         (n = number of contiguous tracks within the
                            same disc unit requested.  If bit 15 of
                            ITRAK = 1, the program is not suspended if
                            tracks are not available; if bit 15 = Ø, the
                            program is suspended until the tracks are
                            available.)
 STRAK    NOP               (RTE stores starting track number here, or -1
                            if the tracks are not available.)
 DISC     NOP               (RTE stores disc logical unit number here.)
 ISECT    NOP               (RTE stores number of sectors/track here.)
```

FORTRAN

To request *n* tracks and set bit 15, add 1ØØØØØB to *n*.

Example (with no suspension):

```
IRCDE = 4
ITRAK = 1ØØØØØB + n
CALL EXEC (IRCDE, ITRAK, ISTRK, IDISC, ISECT)
```

Example (with suspension until tracks available):

```
IRCDE = 4
ITRAK = n
CALL EXEC (IRCDE, ITRAK, ISTRK, IDISC, ISECT)
```

Comments

RTE only supplies whole tracks within one disc. When writing or reading from the tracks (see READ/WRITE EXEC CALL), RTE does *not* provide automatic track switching; the user program is completely responsible for file management within its tracks. RTE will prevent other programs from writing on these tracks, but not from reading out of them.

The program retains the tracks until it releases them, it is aborted, or the RTE System is restarted from the mass storage device.

DISC RELEASE

Purpose

To release some contiguous mass storage tracks which were previously allocated to the program. (See DISC ALLOCATION EXEC CALL.)

Assembly Language

```
          EXT     EXEC
           :
          JSB     EXEC           (Transfer control to RTE)
          DEF     *+ 5(or 3)     (Point of return from RTE)
          DEF     RCODE          (Request code)
          DEF     NTRAK          (Number of contiguous tracks, or -1)
          DEF     STRAK          (Starting track number)
          DEF     DISC           (Mass storage logical unit)
          return point           (Continue execution)
           :
RCODE     DEC     5              (Request code = 5)
NTRAK     DEC     n              (If n = -1, release all tracks assigned
                                 to program, (STRAK and DISC are un-
                                 necessary, so the return point is *+3).
                                 Otherwise, n is the number of contiguous
                                 tracks to be released starting at STRAK.)
STRAK     NOP                    (Starting track number)
DISC      NOP                    (Mass storage logical unit)
```

FORTRAN

Release of *n* Contiguous Tracks Starting at *m* on L.U.*p*:

```
IRCDE = 5
NTRAK = n
ISTRK = m
IDISC = p
CALL EXEC (IRCDE, NTRAK, ISTRK, IDISC)
```

Release all Tracks Allocated to the Program

```
IRCDE = 5
NTRAK = -1
CALL EXEC (IRCDE, NTRAK)
```

PROGRAM COMPLETION

Purpose

To notify RTE that the calling program is finished and wishes to terminate. RTE returns the program to the dormant list.

Assembly Language

```
          EXT    EXEC
           :
          JSB    EXEC     (Transfer control to RTE)
          DEF    * + 2    (Return point from RTE)
          DEF    RCODE    (Request code)
          return point
           :
RCODE     DEC    6        (Request code = 6)
```

FORTRAN

The FORTRAN and ALGOL compilers generate a PROGRAM COMPLETION EXEC CALL automatically when they compile and END statement.

PROGRAM SUSPEND

Purpose

To suspend the calling program from execution until restarted by the GO operator request.

Assembly Language

```
          EXT     EXEC
           :
          JSB     EXEC       (Transfer control to RTE)
          DEF     * + 2      (Point of return from RTE)
          DEF     RCODE      (Request code)
          return  point      (Continue execution)
           :
RCODE     DEC     7          (Request Code = 7)
```

FORTRAN

The FORTRAN (and ALGOL) library subroutine PAUSE, which is automatically called by a PAUSE statement, generates the SUSPEND EXEC CALL.

Comments

When the operator reschedules the program with a GO, the B-Register contains the address of a five-word parameter array set by the GO request. (The parameters equal zero if no values have been given.) In a FORTRAN program, the library subroutine RMPAR can load these parameters; however, the call to RMPAR must occur immediately following the SUSPEND EXEC call, as in the following example:

```
DIMENSION I (5)
CALL EXEC (7)      (Suspend)
CALL RMPAR (I)     (Return point; get parameters)
```

The PROGRAM SUSPEND EXEC CALL is similar to the SS operator request. (See Section III.)

PROGRAM SEGMENT LOAD

Purpose

To load a background segment of the calling program from the disc into the background overlay area and transfer execution control to the segment's entry point. (See Section V, REAL-TIME PROGRAMMING, for information on segmented programs.)

Assembly Language

```
          EXT    EXEC
           :
          JSB    EXEC      (Transfer control to RTE)
          DEF    * + 3     (Point of return from RTE)
          DEF    RCODE     (Request code)
          DEF    SNAME     (Segment name)
          return point     (Continue execution)
           :
RCODE     DEC    8         (Request code = 8)
SNAME     ASC    3,xxxxx   (xxxxx is the segment name)
```

FORTRAN

```
          IRCDE = 8
          DIM INAME (3)
          INAME (1) = xxxxxB    (First two characters)
          INAME (2) = xxxxxB    (Second two)
          INAME (3) = xxxxxB    (Last character)
          CALL EXEC (IRCDE, INAME)
```

<u>Comments</u>

See Section V, OVERLAY SEGMENTS and SEGMENTED PROGRAMS, for a description of segmented background programs.

In the FORTRAN calling sequence, the name of the segment must be converted from ASCII to octal and stored in the INAME array, two characters per word.

PROGRAM SCHEDULE

Purpose

To schedule a dormant program for execution, and optionally, to transfer up to five parameters to that program.

Assembly Language

```
              EXT    EXEC
               :
              JSB    EXEC          (Transfer control to RTE)
              DEF    *+3+ n        (Return point, n = number of parameters)
              DEF    RCODE         (Request code)
              DEF    PNAME         (Name of program to schedule)
              DEF    P1      }
               :             }     (Up to five optional parameters)
              DEF    P5      }
              return point         (Continue execution)
               :
RCODE         DEC    9 (or 10)     (9 = schedule with wait,
                                    10 = no wait; see Comments)
PNAME         ASC    3,xxxxx       (xxxxx is the name of the program to
                                    schedule.)
P1   }
     }                             (Up to five optional parameters)
P5   }
```

FORTRAN

(Can only pass value parameters, not address)

Schedule with Waiting (see *Comments* below)

```
          IRCDE        = 9
          DIM INAME  (3)
          INAME(1)     = xxxxxB ┐
          INAME(2)     = xxxxxB ├  (Name of program in octal;
          INAME(3)     = xxxxxB ┘   see previous request, Comments.)
          CALL EXEC (IRCDE, INAME)
```

Schedule with No Waiting (see *Comments* below)

```
          IRCDE        = 1Ø
          DIM INAME  (3)
          INAME(1)     = xxxxxB ┐
          INAME(2)     = xxxxxB ├  (Name of program in octal)
          INAME(3)     = xxxxxB ┘
          CALL EXEC (IRCDE, INAME)
```

Comments

WAITING AND NO WAITING

A schedule with waiting (RCODE = 9) causes RTE to put the calling program in waiting status. The called program runs at its own priority, which may be greater than, less than, or equal to that of the calling program. Only when the called program terminates does RTE resume execution of the original program at the point immediately following the schedule call.

A background disc-resident program must *not* schedule another background disc-resident program with waiting. RTE aborts the calling program and prints the error message "SCØ3". A real-time disc-resident program, however, may schedule another real-time disc-resident program with waiting, because real-time disc-resident programs are swapped when they conflict over use of their core area.

All other schedule combinations are legal: a disc-resident can call a core-resident, a core-resident can call a disc-resident, and a core-resident can call a core-resident.

A SCHEDULE EXEC call with no waiting (RCODE = 1Ø) causes the specified program to be scheduled for execution according to its priority.

PARAMETERS

When the called program begins executing, the B-Register contains the address of a five-word list of parameters from the calling program (the parameters equal zero if none were specified). A call to the library subroutine RMPAR, the first statement of a called FORTRAN program, transfers these parameters to a specified five-word array within the called program. For example,

```
PROGRAM XQF
DIM IPRAM (5)
CALL RMPAR (IPRAM)
```

The PROGRAM SCHEDULE EXEC CALL is similar to the ON operator request.(See Section III.) The EXECUTION TIME EXEC CALL also schedules programs for execution, but without passing parameters.

TIME REQUEST

Purpose

To request the current time recorded in the real-time clock.

Assembly Language

```
          EXT     EXEC
           :
          JSB     EXEC        (Transfer control to RTE)
          DEF     * + 3       (Point of return from RTE)
          DEF     RCODE       (Request code)
          DEF     ARRAY       (Time value array)
          return point        (Continue execution)
RCODE     DEC     11          (Request code = 11)
ARRAY     BSS     5           (Time value array)
```

FORTRAN

```
          IRCDE = 11
          DIM ITIME (5)
          CALL EXEC (IRCDE, ITIME)
```

Comments

When RTE returns, the time value array contains the time on a 24-hour clock:

ARRAY	or	ITIME (1)	=	Tens of milliseconds
ARRAY + 1	or	ITIME (2)	=	Seconds
ARRAY + 2	or	ITIME (3)	=	Minutes
ARRAY + 3	or	ITIME (4)	=	Hours
ARRAY + 4	or	ITIME (5)	=	Day of the year

The TIME REQUEST EXEC CALL is similar to the TI operator request. (See Section III.)

EXECUTION TIME

Purpose

To schedule a program for execution at specified time intervals, starting after an initial offset time (Initial Time Version) or at a particular absolute time (Absolute Start-Time Version). RTE places the specified program in the time list and returns to the calling program.

Assembly Language (Initial Offset Version)

```
            EXT    EXEC
             :
            JSB    EXEC       (Transfer control to RTE)
            DEF    * + 6      (Point of return from RTE)
            DEF    RCODE      (Request code)
            DEF    PROG       (Program to put in time list)
            DEF    RESL       (Resolution code)
            DEF    MTPLE      (Execution multiple)
            DEF    OFSET      (Initial time offset)
            return point      (Continue execution)
RCODE       DEC    12         (Request code = 12)
          { NOP               (Put calling program in time list)
PROG      {  or
          { ASC    3,xxxxx    (xxxxx is the program to put in the
                               time list.)
RESL        DEC    X          (X is the resolution code.)      See IT,
MTPLE       DEC    Y          (Y is the execution multiple.)   Section III.
OFSET       DEC    -z         (x = units) gives the initial offset; the
                              negative sign signals the call version to RTE.)
```

FORTRAN (Initial Offset Version)

```
        IRCDE      = 12
        IPROG      = Ø or DIM IPROG (3)
                          IPROG (1)=xxxxxB ⎫  (Define program name;
                          IPROG (2)=xxxxxB ⎬  see PROGRAM SEGMENT LOAD
                          IPROG (3)=xxxxxB ⎭  EXEC CALL)
        IRESL      = x ⎫
                       ⎬ (see IT, Section III)
        MTPLE      = y ⎭
        IOFST      =-Z     (Z(x=units)gives the initial offset; the
                           negative sign signals the call version to
                           RTE.
        CALL EXEC (IRCDE, IPROG, IRESL, MTPLE, IOFST)
```

Assembly Language (Absolute Start-Time Version)

The Absolute Start-Time Version is the same as the Initial Offset Version until parameter 5 (OFSET). The sign must be positive. The positive number in parameter 5 signals RTE to expect three more parameters. The return point must be "* + 9" instead of "* + 6". For example,

```
        parameter five  =  DEF HOURS (instead of negative OFSET)
                           DEF MINS
                           DEF SECS
                           DEF MSECS
                           return point
                               :

        HOURS              DEC a ⎫ (Absolute starting time
        MINS               DEC b ⎪ in hours, minutes, seconds,
        SECS               DEC c ⎬ and tens of milliseconds,
        MSECS              DEC d ⎭ on a 24-hour clock)
```

<u>FORTRAN</u> (Absolute Start-Time Version)

```
IRCDE  ⎫
IPROG  ⎬ (Same as Initial Offset Version)
IRESL  ⎪
MTPLE  ⎭
IHRS  = a ⎫ (Absolute starting time
MINS  = b ⎬ in hours, minutes,
ISECS = c ⎪ seconds, and tens of
MSECS = d ⎭ milliseconds on a 24-hour clock)
CALL EXEC(IRCDE,IPROG,IRESL,MTPLE,IHRS,MINS,ISECS,MSECS)
```

Comments

The EXECUTION TIME EXEC call is similar to the IT operator request (see Section III) except that IT only changes the *time values* but does not put the program in the *time list.*

Notice that this call can place either itself *or* another program in the time list, with an execution time cycle starting at a specified real time or starting after a fixed offset from the current time.

ERROR MESSAGES

When RTE discovers an error in an EXEC call, it terminates the program, releases any disc tracks assigned to the program, prints an error message on the operator console, and proceeds to execute the next program in the scheduled list.

When RTE aborts a program, it prints this message:

name ABORTED

When a memory protect violation occurs that is not an EXEC call, this message is printed:

MP *name address* (*address* is the location that caused the violation.)

When an EXEC call contains an illegal request code, this message is printed:

RQ *name address* (*address* is the location that made the illegal call.)

The general error format, for other errors, is:

type name address

where	*type*	is a 4-character error code.
	name	is the program that made the call.
	address	is the location of the call (equal to the exit point if the error is detected after the program suspends).

Error Codes for Schedule Calls

SCØ1	=	Missing parameter
SCØ2	=	Illegal parameter
SCØ3	=	Program cannot be scheduled

SCØ3 INT *XX* occurs when an external interrupt attempts to schedule a program that is already scheduled. RTE ignores the interrupt and returns to the point of interruption. *XX* is the interrupt location address.

SCØ5	=	Program given is not defined
SCØ6	=	No resolution code in EXECUTION TIME EXEC call

Error Codes for Disc Allocation Calls

DRØ1	=	Insufficient number of parameters
DRØ2	=	Number of tracks is zero, >255, or <zero; illegal logical unit; or number of tracks to release is zero or negative.
DRØ3	=	Attempt to release track assigned to another program

Error Codes for I/O Calls

IOØ1	=	Not enough parameters
IOØ2	=	Illegal logical unit
IOØ3	=	Logical unit not assigned
IOØ4	=	Illegal user buffer
IOØ5	=	Illegal disc track or sector
IOØ6	=	Reference to a protected track; or using load-and-go before assigning load-and-go tracks (see LG, Section III)
IOØ7	=	Illegal read or write
IOØ8	=	Disc transfer longer than track boundary
IOØ9	=	Overflow of load-and-go area

SECTION V
REAL-TIME PROGRAMMING

Section V describes the operating procedures and formatting conventions of the seven background programming aids of the Real-Time Software:

- Editor
- Load-and-Go
- FORTRAN Compiler
- Assembler
- Relocating Loader
- Relocatable Library
- ALGOL Compiler

The RTE Editor creates and edits tapes and files of source programs written in FORTRAN And Assembly Language. In load-and-go operations, the RTE FORTRAN Compiler, RTE ALGOL Compiler and RTE Assembler generate relocatable binary code onto temporary disc storage. The RTE Relocating Loader can relocate and merge the code with referenced subroutines of the RTE Relocatable Library and, optionally, make it a permanent part of the RTE System. Once loaded, a program is scheduled for execution by the ON operator request. (See Section III.)

RTE EDITOR

The RTE Editor, a general-purpose background program, creates, lists, and edits symbolic source language tapes and disc files. Only the Editor can create and release source files.

Editor Input and Output Files

In general, RTE Editor requires two inputs: an Edit File containing edit commands, and a Symbolic File containing source programs, on tape or disc. Both the Edit File and the Symbolic File may consist of more than one physical tape.

The RTE Editor processes all properly related inputs and produces a single Updated File on a paper tape or a disc file or a list of the source file. Both the Symbolic File and the Updated File consist of a series of records; each record contains 1 to 72 ASCII characters. In a disc file, records are packed within a track, and tracks are chained together.

Editing Process

The Editor is turned on by an ON, EDIT operator request and reads the Edit File (which it stores in core) from a specified input device other than 2, checking the file for possible format errors. The Editor terminates if available memory is exceeded. After storing the Edit File, the Editor reads the Symbolic File from a specified input device (paper tape or disc), counting each record as entered and performing the required editing before adding the record to the Updated File.

The Editor can also list program tapes or files and create disc files; but in these two cases, it does no editing and does not expect an Edit File.

Operating Procedures

The choice of input device for the Edit File depends on the complexity of the editing. For only one or two short edit commands, using a teleprinter keyboard is faster; but for a longer Edit File, it is faster to punch the file on tape and read it through the photo-reader. The Edit File must be in the photo-reader before scheduling the Editor with an ON operator request. If the edit file is entered through the systems teleprinter, the following message will be printed.

```
/EDIT: ENTER EDIT FILE:
```

SCHEDULING RTE EDITOR WITH "ON"

ON, EDIT, *p1*, *p2*, *p3*, *p4*

where *p1* = the logical unit number of the Edit File input device. The standard input (logical unit 5) is used if none is specified (must not = 2).

p2 = the logical unit number of the Symbolic File input device. The standard input (logical unit 5) is used if none is specified. If *p2* = 2 (disc), the Symbolic File must be a disc file defined by an LS operator request preceding the ON, EDIT request or else the edit aborts (see *LS*, Section III). The old file is released upon completion of editing.

p3 = the logical unit number of an output device for the Updated File or the listing (see *p4*). For a disc file, *p3* = 2.

p4 = the type of Editor operation:

If *p4* = Ø (or no value), a normal edit;
If *p4* = 1, the Symbolic File is only listed (*p3* may not be 2).
If *p4* = 2, a disc file is generated from the Symbolic File (*p3* is ignored). If *p2* = 2, the old file is *not* released.

At the end of the Edit File, RTE Editor prints the message

```
/EDIT:  END EDIT FILE
```

and suspends itself.

The Symbolic File must be loaded into the correct device before typing

```
GO, EDIT
```

to resume operations. If the Updated File is paper tape, RTE Editor generates blank leader on the output tape.

The Editor suspends itself on the End-Of-Tape condition. RTE outputs a message and the I/O device is set down. The device must be declared up before continuing:

```
UP, n
```

where *n* is the decimal EQT entry number of the device.

Editing can continue, or terminate at this point. The operator inputs:

```
GO, EDIT, m
```

where *m*, if Ø (or not given), means to read the next tape;
  if *m* is any other number, it means terminate the editing process.

If the Symbolic File is on the disc, RTE Editor runs to completion without operator intervention. If the Updated File is a paper tape, RTE Editor produces trailer before terminating. If the edit operation is from a disc file to a disc file, the old file is released. If the Updated File is a disc file or if a disc file is created, RTE Editor prints the logical unit numbers and track number of the file in decimal:

```
/EDIT: TRACKS IN NEW FILE:
/EDIT: ln, track 1
/EDIT: ln, track 2
  :
 etc
```

Editing finally terminates with the messages:

```
/EDIT: END OF EDIT RUN
```

RTE Editor must be rescheduled for the next operation.

Edit Commands

The edit commands, consisting of ASCII characters terminated by a RETURN and a LINE FEED, direct the editing process and have this format:

/ ee, p1, p2, p3

where the first non-blank character must be a slash (/),

ee is a one or two-character editing code, and

p1 through *p3* are either record sequence numbers or character numbers referring to the Symbolic File.

Sequence numbers are from one to four digits, character numbers either one or two digits. All sequence numbers must be in ascending order, greater than zero, and unique; a particular sequence number may be used in only one edit command.

EDIT COMMAND FORMATS

Format	Function
/I, *r*	Insert new records after record *r*. The new records follow the /I command in the Edit File.
/D, *r1* [,*r2*]	Delete record *r1*, or records *r1* through *r2*, inclusive.
/R, *r1* [,*r2*]	Replace record *r1*, or records *r1* through *r2*, inclusive. The replacement records follow the /R command.
/CI, *r, c*	Insert new character(s) after character *c* in record *r*. The new characters follow the /CI command.
/CD, *r, c1,* [,*c2*]	Delete character *c1*, or characters *c1* to *c2*, inclusive, in record *r*.

EDIT COMMAND FORMATS (Cont.)

Format	Function
/CR,*r*,*c1*,[,*c2*]	Replace character *c1* to *c2*, inclusive, in record *r*. With the character(s) following the /CR.
/E	Ends the Edit File, or, without other commands, dumps a file from the disc on the output device or copies a tape.
/A	Aborts editing -- useful only when entering the Edit File from teleprinter.

Edit File Editing

Sometimes, a mistake is made near the end of a long Edit File tape. In these cases, the Editor itself can correct the mistakes on the edit tape.

For most commands, the format is the same as regular editing; but if edit commands must be replaced or inserted, the RTE Editor must be able to make a distinction. That is, it has to know whether to execute an edit command, to insert it, or to replace it. For commands to be inserted or replaced, a character "!" makes the distinction. For example,

```
/R, 6
!/D, 314, 315
```

tells RTE Editor to "replace record 6 (in the first Edit File) with the record "/D, 314, 315".

Editor Error Messages

RTE Editor prints error messages on the operator console in this format:

/EDIT: *error message : illegal edit command*

Then the RTE Editor continues with the Edit File; there is no on-line correction of illegal edit commands.

Error Message	Meaning
MEM OVERFLOW	The Edit File overflows available memory; RTE Editor prints the command causing the overflow. Edit terminates.
CS ERR	Illegal edit command, which is printed.
PARAM ERR	Edit command "r" or "c" is illegal: non-numeric, = Ø, >72, $r_2 \leq r_1$, $c_2 \leq c_1$; command printed.
SEQ ERR	"r" parameter ≤ a previous "r", or "r" greater than range of Symbolic File; command printed.
/I ERR	No insert source statements after /I; command printed.
/R ERR	No replacement statements after /R; command printed.
/C OVF	Character overflow in edit statement (i.e., >72 characters)
DISK OVF	No disc space for file; edit terminates.
FILE UN	Undefined Symbolic File, or LUN(Edit File)=2. Edit terminates.

LOAD-AND-GO FACILITY

The Real-Time Executive System provides the facility for "load-and-go" which is defined as compilation or assembly, loading, and executing of a user program without intervening object paper tapes. To accomplish this, the compiler or assembler stores the relocatable object code, which it generates from source statements, on the disc in a predefined group of "load-and-go" tracks. Then separate operator requests initiate loading (ON, LOADR) and execution (ON, *program*).

RTE can store the object code of one main program or one segment and associated subroutines on the disc. The Relocating Loader locates it on the disc, relocates it into an executable absolute program unit, and initializes the load-and-go tracks for further relocatable code (i.e., once the load-and-go information is loaded, it cannot be used again).

The LG operator request (see Section III) initializes and releases the disc tracks required for the load-and-go facility.

RTE FORTRAN COMPILER

The RTE FORTRAN Compiler, a segmented program, executes in the background under control of RTE. The compiler consists of a main program (FTN) and four overlay segments (FTNØ1, FTNØ2, FTNØ3, FTNØ4), and resides in the protected area of the disc. (At least a 4K background disc-resident area is required to execute FORTRAN.)

RTE FORTRAN language, a problem-oriented programming language translated by the compiler, is very similar to regular HP FORTRAN. Source programs, accepted from either an input device or a source file created by RTE Editor, are translated into relocatable object programs, and punched on paper tape and, optionally, stored in the load-and-go tracks of the disc. The object programs can be loaded by the RTE Relocating Loader and executed by an ON operator request (see Section III, *ON*). When a FORTRAN program has been completely debugged, the RTE Relocating Loader can make it a permanent part of the RTE System, if desired.

Compiler Operation

An ON, FTN operator request schedules the RTE FORTRAN compiler for execution; but before the ON, FTN, the operator must place the source program in the input device, or, if input is from a source file, specify the file location with an LS operator request. If planning to load and go, the operator allocates load-and-go tracks with an LG operator request. A sample format for compiler scheduling is:

ON, FTN, *p1*, *p2*, *p3*, *p4*, 99

where

p1	=	logical unit of input device (set to 5 if not given; use 2 for source file input from the disc).
p2	=	logical unit of list device (set to 6 if not given).
p3	=	logical unit of punch device (set to 4 if not given).
p4	=	lines/page on listing (set to 56 if not given).
99	=	the load-and-go parameter. If present, the object program is stored in the load-and-go tracks for later loading. Any punching requested still occurs. (The 99 may occur anywhere in the parameter list, but terminates the list.)

MESSAGES TO OPERATOR DURING COMPILATION

```
I/O ERR ET EQT#
```

This message is printed on the operator console when an end-of-tape occurs on device #*n*. EQT #*n* is unavailable (see *DN*, Section III) until the operator declares it up:

```
UP,n
```

Thus, more than one source tape can be compiled into one program. The next source tape is loaded each time the compiler detects an End-of-Tape.

At the end of compilation, the following message is printed.

```
$END, FTN
```

and RTE executes the next scheduled program.

Two messages are I/O error messages generated by RTE when FTN attempts to write on the load-and-go tracks (RTE aborts FTN).

```
IO∅6
IO∅9
```

IO∅6 means that the load-and-go tracks were not defined (ie., by LG), and IO∅9 means that the load-and-go tracks overflowed. The operator must define more load-and-go tracks with LG and start compilation over again.

The compiler terminates abnormally if

- No source file is declared by LS, although logical unit 2 is given for input. Error E-∅∅19 is printed on the list device.
- The symbol table overflows. (E-∅∅14)
  $END,FTN does not appear after the error message.

RTE FORTRAN Language

The RTE FORTRAN language is similar to the regular HP FORTRAN language. The differences are described in the next few pages. RTE FORTRAN has additional capabilities, using EXEC calls. Read Section IV for complete details on the EXEC calls.

FORTRAN CONTROL STATEMENT

Besides the standard options described in the FORTRAN manual, two new compiler options, T and *n*, are available.

T

Lists the symbol table for each program in the compilation. If a "u" follows the address of a variable, that variable is undefined (the program does not assign a value to it). The A option includes this T option.

n

n is a decimal digit (1 through 9) which specifies an error routine. The user must supply an error routine, ERR*n*. If this option does not appear, the standard library error routine, ERRØ, is used. The error routine is called when an error occurs in ALOG, SQRT,.RTOR, SIN, COS,.RTOI, EXP,.ITOI or TAN.

I/O LOGICAL UNIT NUMBERS

RTE has new function assignments for the logical unit numbers.

Logical Unit Number	Function
1	System Teleprinter
2	System Mass Storage
3	Auxiliary Disc No. 1
4	Standard Punch Device
5	Standard Input Device
6	Standard List Device
7	Can be assigned to any devices by the user, for the defined range of logical units.
8	
9	
10	
:	
63_{10}	

PROGRAM STATEMENT

The program statement, which must be the first statement in a source program, includes optional parameters defining the program type, priority, and time values:

```
PROGRAM name (p1, p2, p3, p4, p5, p6, p7, p8)
```

where *name* is the name of the program (and its entry point),

p1 is the program type (set to 3 for main program, or 7 for subroutines, if not given)

- Ø = System Program
- 1 = Real-Time Core-Resident
- 2 = Real-Time Disc-Resident
- 3 = Background Disc-Resident
- 4 = Background Core-Resident
- 5 = Background Segment
- 6 = Library (re-entrant or privileged)
- 7 = Utility

p2 is the priority (Ø-99, set to 99 if not given)

p3 is the resolution code
p4 is the execution multiple
p5 is hours
p6 is minutes
p7 is seconds
p8 is tens of milliseconds

(Time values are set to Ø if not given. See Section III, IT, for meaning)

DATA STATEMENT

A new statement, the DATA statement, has been added to RTE FORTRAN. DATA sets initial values for variables and array elements. The format of the DATA statement is:

DATA $k_1/d_1/,k_2/d_2/,\ldots,k_n/d_n/$

where k is a list of variables and array elements separated by commas,
d is a list of constants or signed constants, separated by commas and optionally preceded by $j*$ (j is an integer constant).

The elements of d_i are serially assigned to the elements of k_i. The form $j*$ means that the constant is assigned j times. The k_i and d_i must correspond one-to-one.

Elements of k_i may not be from COMMON.

Arrays must be defined (i.e., DIMENSION) before the DATA statements in which they appear.

Example,

```
DIMENSION A(3), I (2)
DATA A (1), A(2), A(3)/1.Ø,2.Ø,3.Ø/, I (1), I (2)/2*1/
```

EXTERNAL STATEMENT

With the new statement, EXTERNAL, subroutines and functions can be passed as parameters in a subroutine or function call. For example, the routine XYZ can be passed to a subroutine if XYZ is previously declared EXTERNAL. Each program may declare up to five EXTERNAL routines.

The format of the EXTERNAL statement is

EXTERNAL $v_1, v_2, \ldots, v_5$

where v_i is the entry point of a function, subroutine, or library program, which exists externally.

EXAMPLE

```
FUNCTION RMX (X,Y,A,B)
RMX=X (A) * Y (B)
END
EXTERNAL XYZ, FL1

Z=Q-RMX (XYZ, FL1, 3.56,4.75)
```

Note: If a library routine, such as SIN, is used as an EXTERNAL, the compiler changes the first letter of the entry point to "%". Special versions of the library routines exist with the first character changed to "%". See RTE Relocatable Library, in this section.

ERROR E-ØØ18 means too many EXTERNALS.

PAUSE & STOP STATEMENTS

PAUSE causes the following message to be printed.

name: PAUSE *xxxx*

Where *name* is the program name, and
xxxx is the octal number given in the PAUSE.

To restart the program, use a GO operator request.
(See Section III, *GO*.)

STOP causes the program to stop after the following message.

name: STOP *xxxx*

Where *name* is the program name, and
xxxx is the octal number given in STOP.

OVERLAY SEGMENTS

Segmented background programs may be written in FORTRAN, but certain conventions are required. A segment must be defined as type 5 in the PROGRAM statement. Segments are initiated by an EXEC call (see Section IV, *PROGRAM SEGMENT LOAD EXEC CALL*). Each segment must make a dummy call to the main program. They must not use data statements. In this way, the proper linkage is established between mains and segments.

Chaining of segments is unidirectional. Once a segment is loaded, execution transfers to it. The segment, in turn may call another segment, but a segment written in FORTRAN cannot return to the main program. All communication between the main program and segments must be through COMMON.

ERRØ LIBRARY ROUTINE

ERRØ, the error print routine referred to under the FORTRAN control statement, prints the following message whenever an error occurs in a library routine:

name: nn xx

Where *name* is the program name,
nn is the routine identifier, and
xx is the error type.

The compiler generates calls to ERRØ automatically.
If the FORTRAN control statement includes an *n* option, the call will be to ERR*n*, a routine which the user must supply.

Check the FORTRAN manual (Chapter 9.9) for the meaning of error codes.

REFERENCE ON FORTRAN

For a complete description of the FORTRAN language, read the FORTRAN programmer's reference manual (02116-9015). Sections 9.5 and 9.8 of that book do not apply to RTE FORTRAN.

RTE ALGOL COMPILER

The HP Real-Time ALGOL compiler is a segmented program requiring 8K of memory. The compiler accepts source programs written according to regular HP ALGOL with some additions and changes.

Compiler Initiation

The following control statement initiates the compiler:

ON,ALGOL,*p1*,*p2*,*p3*,*p4*,*p5*

where: *p1* = Input unit. If not specified, input unit 5 is assumed. If input unit 2 is specified, the source is from disc. Note that the source file must be specified by an "LS" control statement prior to the ON statement.

p2 = List unit. If not specified, list unit 6 is assumed.

p3 = Punch unit. If not specified, punch unit 4 is assumed.

p4 = Number of lines on a page. If not specified, 56 lines per page are assumed.

p5 = Load-and-Go. If p5 = 99, load-and-go is specified. Note that 99 is reserved for load-and-go and its appearence anywhere in the parameter list is interpreted as load-and-go; it terminates the parameter list. To use load-and-go, the ON control statement must be preceded by an "LG" control statement; otherwise the compilation is aborted by an I/O error. Also, if the load-and-go code overflows the number of specified tracks, the Real-Time EXEC aborts the compilation.

Messages To Operator

When the end of the source tape is encountered the following message is output to the system teleprinter:

I/O ERR ET EQT #n

The compiler waits until the following message is input:

UP,n

If source input is indicated to be from the disc (by p1 = 2 in the ON control statement), and the source pointer is not set, the diagnostic

NO SOURCE

is output to the system teleprinter and the compilation ceases.

At the end of a program, a program-termination request is made to the Executive. No message is printed.

In case of a PAUSE statement, the following message is printed:

NAME: PAUSE *XXXX*

where: *NAME* = the program name

XXXX = number which has no significance.

Execution is then suspended. To restart the program, type

GO,*NAME*[*,P1,P2,P3,P4,P5*]

RTE ALGOL Language

The HPAL control statement for RTE ALGOL does not use the Symbol Option S (sense switch control). Also, after the NAM record-name, additional parameters may be specified. The format of the RTE ALGOL control statement is:

HPAL*,s1,s2,s3,s4,"NAM",p1,p2,p3,p4,p5,p6,p7,p8,p9*

where:	*s1 = L*	:	produce source program listing
	s2 = A	:	produce object code listing
	s3 = B	:	product object tape
	s4 = P	:	a procedure only is to be compiled
	"NAM" =	:	program name
	p1 = n	:	a digit from 1 through 9 specifying the error-routine name. A library routine, ERRn, with n = L-9 must be supplied by the user. If this option is not specified, the error-routine name is ERRØ. The error routine is called when an error occurs in the following routines: ALOG, SQRT, .RTOR, SIN, COS, .RTIO, EXP, .ITOI, TAN.

p2 = Program type

p3 = Priority

p4 = Resolution code (0-4)

p5 = Execution multiple (0-999)

p6 = Hours (0-23)

p7 = Minutes (0-59)

p8 = Seconds (0-59)

p9 = Tens of milliseconds (0-99)

Note that the program-name specified in "NAM" must be eclosed in quotation marks, must be a legitimate identifier, and must not contain blanks.

If no symbols are specified (s1 through s4), and if load-and-go is not specified in the ON, ALGOL control statement, the program is compiled but does not produce output other than diagnostic messages.

If there is an error in the control statement, the diagnostic "HPAL????" is on the system teleprinter. The compiler then returns control to the system.

The parameters p1 through p9 must appear in the order shown above but each parameter is optional. A parameter is set to 0 if not specified, except the priority parameter (p3) which is set to 99, the lowest priority.

If s4 is not specified (P), the program type parameter (p2) is set to 3 if not specified by the programmer. If s4 is specified and the program type parameter is not specified by the programmer, it is set to 7.

RTE ALGOL SEGMENTATION

ALGOL programs can be segmented if certain conventions are followed. A segment must be defined as type 5 in the HPAL statement. The segment must be initiated using the PROGRAM SEGMENT LOAD EXEC call from the main or another segment.

To establish the proper linkage between a main program and its segments, each segment must declare themain code a procedure. For example if MAIN is the main program, the following must be declared in each segment:

```
PROCEDURE MAIN; CODE;
```

Chaining of segments is undirectional. Once a segment is loaded, execution transfers to it. The segment, in turn, may call another segment using a EXEC call, but a segment written in ALGOL cannot return to the main program.

RTE ALGOL I/O

The RTE ALGOL I/O statements should specify the proper logical unit numbers for the RTE configuration.

RTE ALGOL ERROR MESSAGES

See the manual *HP ALGOL* (HP 02116-9072) for the meanings of the HP ALGOL compilation time and run time error messages.

RTE ASSEMBLER

The RTE Assembler, a segmented background program that requires at least a 4K background area, operates under control of RTE. The Assembler consists of a main program (ASMB) and six segments (ASMBD, ASMB1, ASMB2, ASMB3, ASMB4, ASMB5), and resides in the protected system area of the disc.

RTE Assembly Language, a machine-oriented programming language, is very similar to regular HP Extended Assembly Language. Source programs, accepted from either an input device or a source file, are translated into absolute or relocatable object programs; absolute code is punched in binary records, suitable for execution outside of RTE. ASMB can store relocatable code in the load-and-go area of the disc for on-line execution, as well as punch it on paper tape. The RTE Relocating Loader accepts assembly language relocatable object programs from paper tape or the load-and-go area.

The source tape passes through the input device only once, unless there is no disc stroage space. In this case, two passes are required. (See next page *Messages During Assembly*.)

Assembler Operation

An ON operator request schedules the RTE Assembler for execution at its priority level; but before the ON, the operator must place the source program in the input device, or if inputting from a source file, must specify the location with an LS operator request. If planning to load-and-go, the operator must allocate load-and-go tracks with an LG operator request.

```
ON, ASMB, p1, p2, p3, p4, 99
```

where *p1*	=	logical unit of input device (set to 5 if not given; use 2 for source file input from the disc).
p2	=	logical unit of list device (set to 6 if not given).
p3	=	logical unit of punch device (set to 4 if not given).
p4	=	lines/page on listing (set to 56 if not given).
99	=	load-and-go parameter. If present, the object program is stored on the disc for loading, and any punching requested still occurs (the 99, which may appear anywhere in the parameter list, terminates the list).

MESSAGES DURING ASSEMBLY

The messages described in this section are printed at the teleprinter console or in the program listing.

When an end-of-tape occurs on device #*n*, this message appears:

```
I/O ERR ET EQT #n
```

EQT #*n* is unavailable (see DOWN, Section III) until the operator declares it up (see UP, Section III).

```
UP, n
```

Thus, more than one source tape can be assembled into one program. The next tape is loaded each time the input device goes down.

The following message signifies the end of assembly:

```
$END ASMB
```

RTE goes on to execute the next scheduled program.

If another pass of the source program is required, the message appears at the end of pass one.

```
$END ASMB PASS
```

The operator must replace the program in the input device and type:

```
GO, ASMB
```

If an error is found in the Assembler control statement, the following message appears:

```
$END ASMB CS
```

The current assembly stops.

If an end-of-file condition occurs before an END statement is found, the teleprinter signals:

```
$END ASMB XEND
```

The current assembly stops.

If source input for logical unit 2 (disc) is requested, but no file has been declared (see *LS*, Section III), the teleprinter signals:

```
$END ASMB NPRG
```

Assembly stops. RTE generates two messages when ASMB attempts to write on the load-and-go tracks (ASMB aborted):

```
IOØ6
IOØ9
```

IOØ6 means that the load-and-go tracks were never defined by an LG operator request, and IOØ9 means that the load-and-go tracks have overflowed.

The next message is associated with each error diagnostic printed during pass 1.

nnn

nnn is the "tape" number where the error (reported on the next line of the listing) occurred. A program may consist of more than one tape. The tape counter starts with one and increments whenever an end-of-tape condition occurs (paper tape) or a blank card is encountered. When the counter increments, the numbering of source statements starts over at one.

Each error diagnostic printed during pass 2 of the assembly is associated with a different message:

PG *ppp*

ppp is the page number (in the listing) of the *previous* error diagnostic. PG ØØØ is associated with the first error in the program.

These messages (#*nnn* and PG *ppp*) occur on a separate line, just above each error diagnostic in the listing.

RTE Assembly Language

The RTE Assembly Language is equivalent to extended assembly language, as defined in the *ASSEMBLER* programmer's reference manual (02116-9014). A few language changes are required to run under RTE; programs must request certain functions, such as I/O, from the executive. These requests are made using the EXEC calls described in Section IV.

ASSEMBLER CONTROL STATEMENT

The control statement has the same form as that of regular assembly language; and although only relocatable code can be run under RTE, the RTE Assembler accepts and assembles absolute code. Absolute code is never stored in the load-and-go area. To get absolute code, the control statement must include an "A". The "R", however, is not required for relocatable code. An "X" causes the assembler to generate non-extended arithmetic unit code.

NAM STATEMENT

The NAM statement, which must be the first statement in a source program, includes optional parameters defining the program type, priority, and time values:

NAM *name, p1,p2,p3,p4,p5,p6,p7,p8*

where *name* is the name of the program,

p1 is the program type (set to Ø if not given):

- Ø = System Program
- 1 = Real-Time Core-Resident
- 2 = Real-Time Disc-Resident
- 3 = Background Disc-Resident
- 4 = Background Core-Resident
- 5 = Background Segment
- 6 = Library (re-entrant or privileged)
- 7 = Utility

p2 is the priority (Ø to 99, set to 99 if not given)

p3 is the resolution code
p4 is the execution multiple
p5 is hours
p6 is minutes
p7 is seconds
p8 is tens of milliseconds

} (Time values, set to Ø if not given; see Section III, *IT*, for meaning)

These parameters are optional; but if any one parameter is given, those preceding it must appear also.

I/O LOGICAL UNIT NUMBERS

RTE has different function assignments for the logical unit numbers; check *RTE FORTRAN LANGUAGE,* this section, for definitions of logical unit numbers.

ORB STATEMENT

RTE Assembly Language does not contain the ORB statement, since information cannot be loaded into the protected base page area by user programs. However, programs can read information from base page using absolute address operands up to 1777_8.

The memory protect feature, which protects the resident executive from alteration, interrupts the execution of a user program under these conditions:

- Any operation that would modify the protected area or jump into it.
- Any I/O instruction, except those referencing the switch register or overflow.
- Any halt instruction.

Memory protect gives control to RTE when an interrupt occurs, and RTE checks whether it was an EXEC call. If not, the user program is aborted.

Segmented Programs

Background disc-resident programs may be structured into a main program and several overlapping segments, as shown in Figure 5-1. The main program begins from the start of the background disc-resident area. The area for overlay segments starts immediately following the last location of the main program. The segments reside permanently on the disc, and are read in by an EXEC call when needed. Only one segment may reside in core at a time.

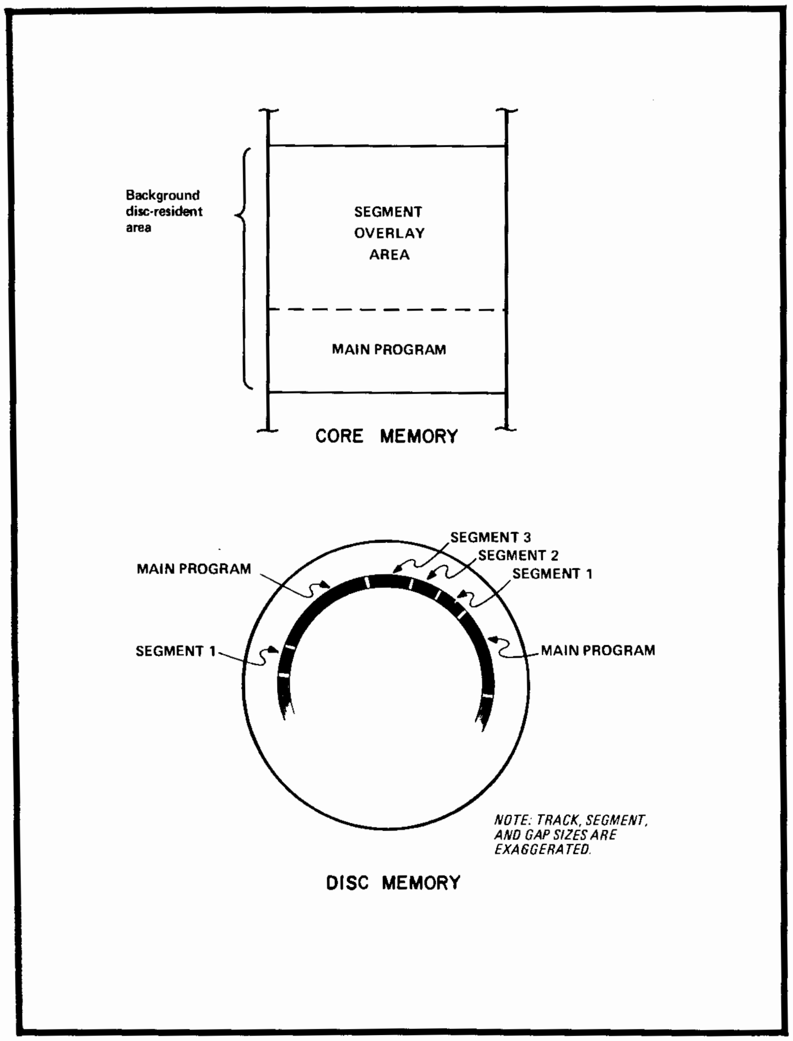

Figure 5-1. Segmented Programs

The main program must be type 3, and the segments must be type 5. When using RTGEN to configure the system, the main program must be entered prior to its segments. One external reference from each segment to its main program is required for RTGEN to link the segments and main programs. Also, each segmented program should use unique external reference symbols. Otherwise, RTGEN may link segments and main programs incorrectly.

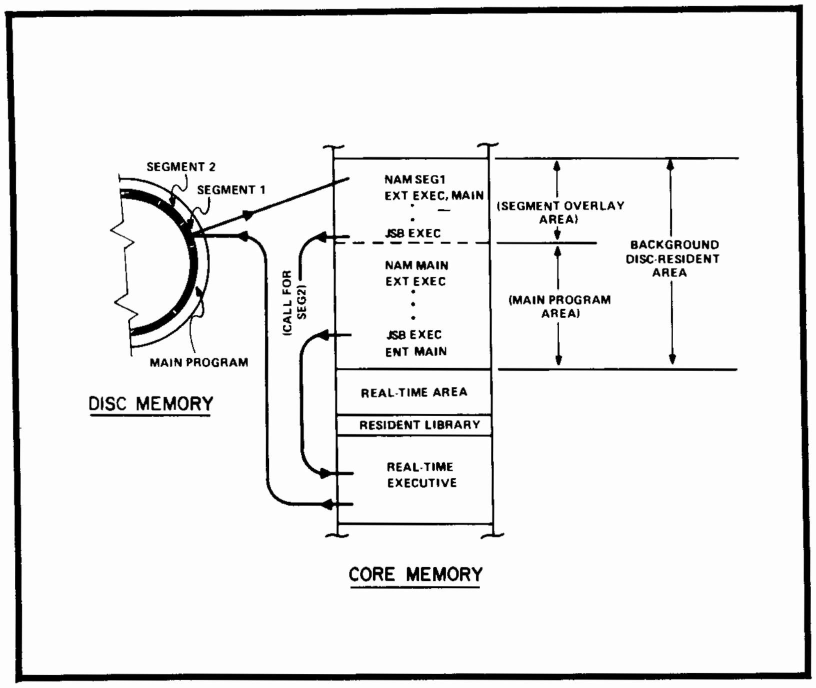

Figure 5-2. Main Calling Segment

Figure 5-2 shows how an executing main program may call in any of its segments from the disc, via a "JSB EXEC". The main program is not suspended, but control is passed to the transfer point of the segment.

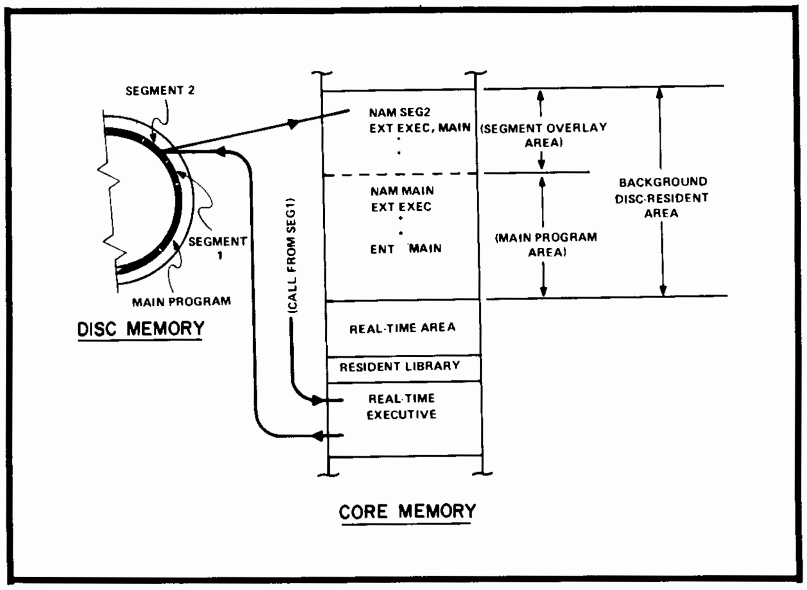

Figure 5-3. Segment Calling Segment

An executing segment may itself call in another of the main program's segments using the same "JSB EXEC" request. (See Figure 5-3.) However, a segment of the FORTRAN or ALGOL Compiler may not call in a segment of the Assembler.

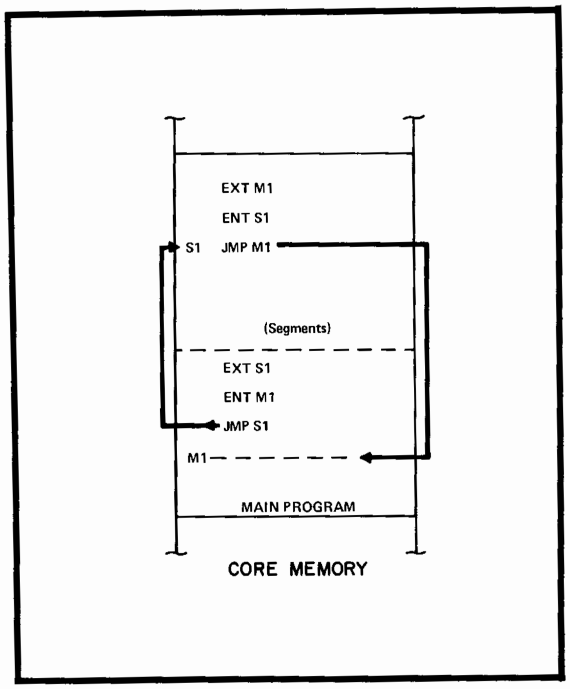

Figure 5-4. Main-to-Segment Jumps

When a main program and segment are currently residing in core, they operate as one single program. Jumps from a segment to a main program (or vice versa) can be programmed by declaring an external symbol and referencing it via a JMP instruction. (See Figure 5-4.) A matching entry symbol must be defined at the destination in the other program. RTGEN associates the main programs and segments, replacing the symbolic linkage with actual absolute addresses (i.e., a jump into a segment is executed as a jump to a specific address). The programmer should be sure that the correct segment is in core before any JMP instructions are executed.

Reference on Assembly Language

Consult the ASSEMBLER programmer's reference manual (02116-9014) for a full description of assembly language. Sections 5.5 and 5.6 of that text do not apply to RTE.

RTE RELOCATING LOADER

The RTE Relocating Loader, a background disc-resident program, has two basic functions:

- To relocate and load background programs (without incorporating them permanently into the system) so that they can be scheduled by an ON operator request.
- To permanently modify--by adding or replacing--the set of disc-resident programs, both real-time and background.

The loader accepts programs that are generated by the RTE FORTRAN Compiler, RTE ALGOL Compiler and RTE Assembler, and transmitted to the loader on paper tape or through load-and-go tracks on the disc.

This section describes background loading first, then on-line modification.

Background Loading

Programs, including segmented programs, can be loaded from a paper tape or the load-and-go tracks of the disc. After loading, the programs are relocated to the start of the background disc-resident area and linked to external references (such as EXEC, the resident library, or the relocatable library). Any segments overlay the core area following the main program and its subroutines.

These programs, regardless of the program type recorded in the NAM record, are treated as background disc-resident programs but are *not* permanently added to the protected RTE System (as is done in on-line modification to the system which will be explained later). The DEBUG library subroutine can be linked to the program if desired. (See *Starting the Loader-Background Operations*.)

The loader stores the absolute version of the program and its subroutines and linkages on a disc track or a group of contiguous tracks; it then assigns the disc tracks to the RTE System (i.e., not available as scratch or data tracks by programs) and updates, in core only, the ID segment assigned to the program. The program and its subroutines may be as large as the background disc-resident area, except that any common region is allocated within this area, not the regular background common area.

Starting the Loader - Background Operations

The operator schedules the RTE Relocating Loader for execution with an ON, LOADR request.

ON,LOADR,*p1*,*p2*,*p3*,*p4*,*p5*

where *p1* = logical unit number of input device (if set to Ø, 5 is used; if set to 99, the load and go tracks are used).

p2 = logical unit number of the list output device for printing loading information (if set to Ø, 6 is used).

p3 = operation code:

Ø - normal background load

1 - background load with DEBUG

(2 is on-line modification, 3 is list all programs)

p4 = structure parameter:

Ø - single main program and subroutines

1 - main and segments

p5 = if 1, omit list of program names and locations during loading.

If any parameter is missing, a zero value is assumed for the parameter.

COMMENTS

Selecting operation code 1 (*p3*) causes DEBUG to be appended to each main program and segments. The loader sets the primary entry point of each to DEBUG, rather than the user program. When the program is run, DEBUG takes control of program execution and requests instructions from the keyboard. (See "DEBUG" for legal DEBUG commands.)

If the structure parameters of the ON,LOADR operator request (*p4*) equal zero, a single main program and subroutines will be merged into an absolute program unit. To load another main program, the loader must be scheduled again.

If the structure parameter (*p4*) equals one, a main program and segments will be loaded. However, only one part of the segmented program may reside on the load-and-go tracks of the disc; the others must be on paper tape.

After the loader is initiated, there is a perceptible delay while the loader allocates a disc track (or three contiguous tracks for segmented programs) and writes zeroes in every location. Thus, when the program is stored on the disc, any BSS location will contain NOP instructions (i.e. zero).

Entering the Relocatable Object Code

For a main/segment load, the main program must enter first to establish the segment area boundaries. The library must be scanned after each main program and segment.

The loader scans the relocatable programs and subroutines as it reads them in, keeping track of any external references. If input is initially from the disc as specified by 99 in the ON statement, the loader immediately scans the library for entry points. If input is from paper tape or from the disc, the loader suspends with the message "LOAD" when it finds an end-of-tape mark. The operator reschedules the loader with this GO request.

```
                    GO,LOADR,p1[,p2]

where   p1  =  load option parameter:
                    Ø - load from the input unit.
                    1 - load subroutines needed from the
                        relocatable library.
                    2 - load subroutines from the load-and-
                        go area of the disc (the library is not
                        automatically scanned after this operation).
                    3 - load from the relocatable library
                        for the last segment in a main/
                        segment load.
        p2  =  1, if present, omit the list of entry points at
                  the end of loading.
```

Matching Entries and Externals

External references to resident library programs use the existing base page links to those entry points, but external references to disc-resident relocatable library subroutines cause these routines to be loaded along with the referencing program. If a segment references a library routine also referenced by the main program, the segment shares the routine loaded with the main program.

After matching all possible entry points to external references, if there are still undefined external references, the loader prints this message:

```
UNDEFINED EXTS
```

The external references are listed, one per line, and the loader suspends itself.

To load additional paper tapes, the operator types:

```
GO,LOADR
```

To continue, without fulfilling external references, the operator types:

```
GO, LOADR, 1
```

The loader proceeds to relocate the program or segment and subroutines into absolute format, and prints a list of all entry points (unless instructed not to print the list) as each routine is loaded. The entry point listing format is:

**name address*

where *name* is the entry point name, and
address is its absolute location in octal.

End of Loading

At the end of a normal load, or after loading the last segment of a main/segment load, the loader prints this message:

name READY-LOADING COMPLETE

where *name* is the name of the main user program. The loader then terminates.

After loading a main or segment of a main-segment load, the loader prints this message:

```
LOAD
```

and returns to the "enter program" phase for the next segment.

After entering the *last* segment and subroutines from the input device (not the disc), the operator reschedules the loader with a

```
GO,LOADR,3
```

and the loader proceeds to the end of loading, as above.

The operator can schedule the program for execution by an ON,*name* operator request (see Section III, ON). The disc tracks containing the program are assigned to the system and are software-protected. The program can be eliminated from the system by the OF,*name*,8 operator request (see Section III, OF).

Sample Loading Operation

Following is a special listing from a load operation (operator requests are shown in *SCRIPT*, loader output in OCR):

```
*ON,LOADR
CHK2    30000 30207
I/O ERR ET EQT #04
/LOADR: LOAD
*UP,4
*GO,LOADR,1
FRMTR   30260 33010
%IN     33011 33022
%OS     33023 33034
%AN     33035 33046
  :
.FLUN   34532 34546
PAUSE   34547 34670
ERRO    34671 34737
ENTRY POINTS
 *EXEC    02000
 *$LIBR   02164
 *$LIBX   02267
 *CHK2    30000
    :
 *.IENT   33357
 *.CHEB   34232
 *ARCTA   33420
 *.PAUS   34547
/LOADR: CHK2 READY -LOADING COMPLETE
```

RTE DEBUG Library Subroutine

DEBUG, a utility subroutine of the RTE Relocatable Library, allows programs to be debugged (i.e., checked for logical errors on-line) during execution. Programs that expect starting parameters or that call RMPAR (see Section IV, PROGRAM SUSPEND EXEC CALL) cannot use DEBUG because DEBUG uses the parameters.

The first parameter of the ON request specifies a teleprinter logical unit (DEBUG uses 1, if none is given).

The operator schedules the loader for execution with the usual ON,LOADR request:

ON,LOADR,*p1*,*p2*,*p3*,*p4*,*p5*

If the *p3* parameter of the ON, LOADR operator request equals one, the loader combines DEBUG with the main program and segments being loaded. The primary entry point (the location where execution begins) is set to DEBUG so that when the program is turned on with an ON operator request, DEBUG takes control and prints a message:

```
BEGIN 'DEBUG' OPERATION
```

The programmer can enter any legal debug operation. Illegal requests are ignored and a message is printed:

```
ENTRY ERROR
```

For further details on the uses of DEBUG, refer to the BASIC CONTROL SYSTEM manual (02116-9017) where the non-RTE DEBUG routine is described.

DEBUG OPERATIONS

B,*A*	Instruction breakpoint at address *A*. (NOTE: if *A* = JSB EXEC, a memory protect violation occurs.)
D,A,*N1*[,*N2*]	ASCII dump of core address *N1* or from *N1* to *N2*.
D,B,*N1*[,*N2*]	Binary dump of core address *N1* or from *N1* to *N2*.
M,*A*	Sets absolute base of relocatable program unit.
R,*A*	Execute user program starting at *A*. Execute starting at next location in user program (used after a breakpoint or to initiate the program at the transfer point in the user program).
S,*A1*,*D1*	Set *D1* in location *A1*.
S,*A1*,*D1*...*Dn*	Set *D1* to *Dn* in successive memory locations beginning at location *A1*.
W,A,*D1*	Set A-Register to *D1*.
W,B,*D2*	Set B-Register to *D2*.
W,E,*D3*	Set E-Register (Ø=off, non-zero = on).
W,O,*D4*	Set Overflow (Ø=off, non-zero = on).
X,*A*	Clear breakpoint at address *A*.
A	Abort Debug operation.

Listing the Programs in the RTE System

The operator obtains a listing of all the programs and blank ID segments in the system when parameter *p3* of the ON,LOADR operator request equals three (*p3* = 3, see STARTING THE LOADER - BACKGROUND OPERATIONS). The listing is headed by

SYSTEM PROGRAM LIST: NAME, TYPE, PRIORITY

For each ID segment in the system, this line is printed:

name t pr

where *name* is the program name

t is the program type:

1 - real-time resident

2 - real-time disc-resident

3 - background disc resident

4 - background resident

5 - background segment

pr is the program priority, from 1 to 99.

A blank (i.e., available for use by the loader) ID segment is noted by the line:

<BLANK ID SEGMENT>

The loader terminates after the list is complete.

On-line Modification

Using the RTE Relocating Loader, the operator can permanently modify the set of disc-resident user programs in a configured RTE System. The loader adds new disc-resident real-time or background programs, and replaces disc-resident programs with updated versions that have the same name. The OF operator request (see Section III) deletes those disc-resident programs loaded into the background by the loader.

When the RTE System is generated, RTGEN, the system generator, stores the programs on the disc in an absolute, packed format. Each main program is identified and located by an identification segment (ID segment) in the core resident RTE Area. The program's disc location, only for disc-resident as well as its core memory bounds, is kept in the ID segment.

RTGEN can create a number of blank ID segments so that the loader can add new programs to the permanent system later. The addition or replacement of a program involves the conversion of relocatable programs into an absolute unit, finding space on the disc to store it, and recording information in the ID segment.

In replacing, the new program may overlay the old program's disc space only if the length of the new program plus base page linkages does not exceed the disc space. A track or group of tracks is allocated for the program storage if the space requirements of the new program exceed those of the old, or if simply adding. These tracks are software-protected, but not hardware-protected.

Core-resident programs cannot be replaced because the length of the program and linkage area is not kept in the ID segment for core-resident programs. Core-resident programs cannot be added because this would require changing the disc-resident program area origins.

LIMITATIONS

Several limitations may prohibit the final addition or replacement of disc-resident programs:

- A common length exceeds the original common block.
- The base page linkages exceed the original linkage provided.
- The length of the absolute program unit exceeds the area available.
- Disc space is not available to store the program.
- A blank ID segment is not available for adding a program. (A program previously added could be deleted to provide a blank ID segment.)

The disc hardware protect must be physically disabled prior to the loading (and then enabled afterwards), unless the protection is always kept disabled. RTE provides additional software protection for any tracks containing systems programs or user programs.

ON-LINE MODIFICATIONS: SEGMENTED BACKGROUND PROGRAMS

Segmented programs can be added and replaced using the loader as long as the main program is always entered first.

When replacing segmented programs that were incorporated into the RTE System at generation time, there must be a one-to-one name equivalence between the old and new programs. That is, the operator must replace every segment with a new segment having the same name, or eliminate the segment with an OF,*name*,8 operator request (see Section II). Extra segments, however, may be added in a replacement.

If enough blank ID segments exist for all the parts of the new segmented program, the original ID segments are blanked, but kept for later use of the original disc space allocated to them. If there are an insufficient number of blank ID segments for the new program, it uses the ID segments of the program being replaced and the original disc space is lost.

When replacing segmented programs that were added on-line, there must always be a complete one-to-one name replacement. In no case should OF be used to delete segments. The complete set of disc tracks allocated to the old program is released immediately, and the ID segments are blanked for further use by any program, including the new segmented program.

NEW PROGRAM ADDITION

When a new program is added, it is stored on a complete disc track or several contiguous tracks. A blank ID segment is allocated to record the program's memory and disc boundaries, name, type, priority, and time values (as done by RTGEN; see Section VII). The loader attempts to use available disc space in the system before allocating new full tracks. If new tracks must be allocated, they are assigned to the system and are software-protected.

A program added to the system must be thoroughly debugged because, once incorporated, it has all the rights of an original program. Specifically, a real-time disc-resident program has access to the real-time resident area, the block of system available memory for I/O buffers and re-entrant blocks, and the background areas.

PROGRAM REPLACEMENT

In a replacement, if the new program can fit in the disc area of the old program (both programs must have the same name), the new program uses the ID segment of the old. The new program is generated onto temporary tracks, and then, if it can fit in the old area, it is transferred. If not, it stays on the temporary tracks and a blank ID segment is assigned to it. The old ID segment is blanked but retains its disc space for later use by another program.

ON-LINE MODIFICATIONS: OPERATING PROCEDURES

ON,LOADR,*p1*,*p2*,*p3*,*p4*,*p5*

The operator schedules the loader for on-line modifications to the system by setting parameter *p3* of the ON,LOADR operator request equal to 2 (*p3* = 2, see STARTING THE LOADER - BACKGROUND OPERATIONS).

The loader requires additional information to carry out the modifications, so it prints the message:

```
/LOADR: "GO" WITH EDIT PARAMETERS
```

and suspends itself. The operator, after checking that the hardware disc protect is disabled, reschedules the loader with a GO operator request containing these parameters:

GO, LOADR,*p1*,*p2*[,*p3*]

where *p1* = 1 for an addition operation, or 2 for a replacement operation;

p2 = 2 for a real-time disc-resident program, or 3 for a background disc-resident program;

p3 = priority (optional) from Ø to 99 (a Ø means use the priority value in the NAM record of the program or, if that priority is Ø, use 99).

Any errors cause the error message "L1Ø" to be printed. The disc hardware protect *must* be disabled, and the loading process proceeds exactly as described under background loading. When loading is over, the disc should be hardware-protected again.

Relocating Loader Error Messages

Error messages are printed in this format:

/LOADR: *message*

LØ1 and LØ2 (checksum error and illegal record) are recoverable errors. The offending record can be reread by repositioning the tape and typing:

GO,LOADR

For irrecoverable errors, one of the following messages is printed, followed by "LOADING ABORTED" (the loader is terminated):

LØ3 - Memory overflow
LØ4 - Base page linkage area overflow
LØ5 - Symbol table area overflow
LØ6 - Common block error
- Exceeding allocation in a replacement or addition.
- In a normal background load, first program did not declare largest common block.

LØ7 - Duplicate entry points
LØ9 - Record out of sequence

LØ8 means there was no transfer address (main program) in the program unit. Another program may be entered with a GO operator request. (This also occurs when load-and-go is specified, but no program exists in the load-and-go area.)

L1Ø also means that an error was found in an operator request parameter. GO requests may be retyped; ON requests may not.

ADDITIONAL MESSAGES

"NO BLANK ID SEGMENTS" is printed when an available (i.e., blank) ID segment is not found. The loader calls for program suspension. The operator may then delete a program from the system (OF operator request) or may terminate the loader.

"WAITING FOR DISC SPACE" is printed when a track allocation cannot be made. The loader repeats the disc request and is suspended until space becomes available. This message is primarily for information, as no action is required.

"UNDEFINED EXTS" is printed followed by a list of all remaining undefined external symbols after a scan of the library. Additional programs may be loaded by the GO operator request.

"LOAD" is printed and the loader is suspended whenever an end-of-tape condition is detected from the input unit.

"DUPLICATE PROG NAME - *name*" is printed when a program *name* is already defined in the system for a normal load or a program addition. The loader changes the name of the current program by replacing the first two characters with "##".

THE RELOCATABLE LIBRARIES

There are two libraries or collections of relocatable subroutines that can be used by RTE: the RTE/DOS Relocatable Library (EAU or Non-EAU versions) and the RTE/DOS FORTRAN IV Library. These libraries contain mathematical routines such as SIN and COS, and utility routines such as BINRY, etc. A program signifies its need for a subroutine by means of an "external reference". External references are generated by EXT statements in assembly language, by CALL statements and the compiler in FORTRAN, and by CODE procedures and the compiler in ALGOL.

When the system is generated, several combinations of libraries are possible. Every system must contain an RTE/DOS Relocatable Library: either an EAU version or a non-EAU version, depending on the computer hardware. This library does not contain a formatter, but the FORTRAN IV Library contains a formatter that handles extended precision numbers. If extended precision arithmetic is not needed, a separate RTE/DOS Basic FORTRAN Formatter is available to take the place of the FORTRAN IV Library.

All of these libraries and the subroutines they contain are documented in the *Relocatable Subroutine Manual* (02116-91780).

Re-entrant Subroutine Structure

Many executing programs can reference one resident library subroutine on a priority basis. If the subroutine is structured as re-entrant, it must not modify any of its own instructions; and it must save temporary results, flags, etc., if it is called again (by a higher priority program) before completing its current task.

Each time the re-entrant routine begins executing, the address and length of its temporary data block are transferred to RTE which saves this data. At the end of execution, the re-entrant routine calls RTE again to restore the temporary data, if any. Two entry points in RTE, $LIBR and $LIBX, do the saving and restoring.

Re-entrant structure is used for programs with an execution time exceeding one millisecond. For shorter execution times, the overhead time RTE uses in saving and restoring temporary data makes re-entrant structure unreasonable. Faster subroutines can be of *privileged* structure.

FORMAT OF RE-ENTRANT ROUTINE

```
ENTRY NOP               (Entry point of routine)
      EXT $LIBR,$LIBX
      JSB $LIBR         (Call RTE to save temporary data)
      DEF TDB           (Address of temporary data)
       .
       .
       .                (Program instructions)
EXIT  JSB $LIBX         (Call RTE to restore data)
      DEF TDB
      DEC N             (N is for routines with two return points
                        in the calling program; Ø specifies the
                        error-print return and 1 the normal return.)
                        For routines with only one return point, N=Ø.
TDB   NOP               (Linkage address to previous block)
      DEC N             (Total length of block)
      NOP               (Return address to calling program)
       .                (Temporary data)
       .
```

Privileged Subroutine Structure

Many programs use one copy of a privileged subroutine without incurring re-entrant overhead because privileged subroutines execute with the system interrupt disabled. As a result, privileged routines need not save temporary data blocks but must be very quick in execution to minimize the time that the interrupt system is disabled.

FORMAT OF PRIVILEGED ROUTINE

```
ENTRY NOP               (Entry point to the routine)
      EXT $LIBR,$LIBX
      JSB $LIBR         (Call RTE to disable the interrupt
                        system)
      NOP               (Denotes privileged format)
       .
       .
       .
EXIT  JSB $LIBX         (Call RTE to return to calling
                         program and enable interrupts)

      DEF ENTRY         (Location of return address)
```

Utility Subroutine Structure

Utility subroutines are subroutines which cannot be shared by several programs because of internal design or I/O operations. A copy of the utility routine is appended to every program that calls for it. The library subroutine FRMTR, which carries out FORTRAN I/O operations, and the PAUSE subroutine are utility routines.

When RTGEN creates the disc-resident RTE System, all library subroutines not included in the resident library are stored on the disc in relocatable format as utility routines. They are appended onto programs by the RTE Relocating Loader in its background loading process (see *RTE RELOCATING LOADER*).

SECTION VI
REAL-TIME INPUT/OUTPUT

In the Real-Time Executive System, centralized control and logical referencing of I/O operations effect simple, device-independent programming. Each I/O device is interfaced to the computer through one or more I/O channels which are linked by hardware to corresponding core locations for interrupt processing. By means of several user-defined I/O tables, self-contained multi-device drivers, and program EXEC calls, RTE relieves the programmer of most I/O problems.

For further details on the hardware input/output organization, consult Volume One Specifications and Basic Operation Manual, Model 2116B Computer (02116-9152).

SOFTWARE I/O STRUCTURE

An Equipment Table records each device's I/O channels, driver, DMA, and buffering specification. A Device Reference Table assigns a logical unit number to each entry in the Equipment Table, thus allowing the programmer to reference changeable logical units instead of fixed physical units.

An Interrupt Table keeps track of RTE's activity when an interrupt occurs on each channel: RTE can call a driver, schedule a specified program, or handle the interrupt itself.

Drivers are responsible for initiating and continuing operations on all devices of an equivalent type.

The programmer requests I/O by means of an EXEC call in which he specifies only the logical unit, control information,buffer location, buffer length, and type of operation.

The Equipment Table

The Equipment Table (EQT) has an entry for each device recognized by RTE (these entries are established by the user when the RTE System is generated). The EQT entries reside in the permanent core-resident part of the system, and have this format:

CONTENTS

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

WORD	CONTENTS				
1	Device Suspended List Pointer				
2	Driver "Initiation" Section Address				
3	Driver "Completion" Section Address				
4	D	B	(not used)	Unit #	Channel #
5	AV	EQUIP. TYPE CODE	STATUS		
6	CONWD (Current I/O Request Control Word)				
7	Request Buffer Address				
8	Request Buffer Length				
9	Temporary or Disc Track #				
1∅	Temporary or Starting Sector #				
11	Temporary Storage for Driver				

where

D = 1 if DMA required.

B = 1 if automatic output buffering used.

Unit = sub-channel address, optional.

Channel = I/O select code for device (lower number if a multi-board interface).

AV = availability indicator:

∅ = available for use.

1 = disabled(down).

2 = busy (currently in operation).

3 = waiting for an available DMA channel.

STATUS = the actual physical status or simulated status at the end of each operation. For paper tape devices, two status conditions are simulated:

Bit 5 = 1 means End-Of-Tape on input, or tape supply low on output.

EQUIP, TYPE CODE = type of device. When this number is concatenated with "DVR", it identifies the device's software driver routine:

ØØ to Ø7$_8$ =	paper tape devices,
ØØ =	teleprinter
Ø1 =	photo-reader
Ø2 =	paper tape punch
1Ø to 17 =	unit record devices,
1Ø =	plotter
12 =	line printer
2Ø to 37 =	magnetic tape/mass storage devices,
3Ø =	disc or drum
4Ø to 77 =	instruments.

CONWD = user control word supplied in the I/O EXEC call (see Section IV).

When RTE initiates or continues an I/O operation, it places the addresses of the EQT entry for the device into the Base Page Communication Area (see Appendix A) before calling the driver routine.

Logical Unit Numbers

Logical unit numbers from 1$_{1Ø}$ to 63$_{1Ø}$ provide logical addressing of the physical devices defined in the EQT; these numbers are maintained in the Device Reference Table (DRT), which is created by RTGEN and can be modified by the LU operator request.

Each one-word entry in the DRT contains the EQT entry number of the device assigned to the logical unit. The functions of logical units 1 through 6$_{1Ø}$ are predefined in the RTE System:

1 - System Teleprinter
2 - System Mass Storage
3 - Auxiliary Mass Storage
4 - Standard Punch Unit
5 - Standard Input Unit
6 - Standard List Unit

Logical units 7 through 63 may be used for any functions desired. The operator can assign EQT entries to them when the RTE System is generated (see Section VII) or after the system is running (see Section III, *LU*). The user determines the number of logical units when the system is generated.

Logical unit numbers are used by programs executing in the background or real-time areas to specify on what type of device I/O transfers will be carried out. In an I/O EXEC call, the program simply specifies a logical unit number and does not need to know which actual device or which I/O channel handles the transfer.

The Interrupt Table

The interrupt table contains an entry, established at system generation time, for each I/O channel in the computer.

If the entry is equal to Ø, the channel is undefined in the system. If an interrupt occurs on one of these channels, RTE prints this message:

ILL INT *xx*

where *xx* is the octal I/O channel number. RTE then clears the interrupt flag on the channel and returns to the point of interruption.

If the contents of the entry are positive, the entry contains the address of the EQT entry for the device on the channel. If the contents are negative, the entry contains the negative of the address for the ID segment of a program to be scheduled whenever an interrupt occurs on the channel.

The interrupt locations in core contain a JSB $CIC; CIC is the central interrupt control routine which examines the interrupt table to decide what action to take. On a power failure interrupt RTE halts (however, the user can write his own routine to handle power failure interrupts). If privileged interrupt processing is included in the system, the privileged channels bypass $CIC and the interrupt table entirely.

Input/Output Drivers

The I/O driver routines, part of the resident RTE, handle the actual transfer of information between the computer and external devices. When a transfer is initiated, RTE places the EQT entry addresses into the Base Page Communication Area and jumps to the driver entry point. The driver configures itself for the particular channel (in this way the same driver can handle devices of the same type on many channels), initiates the transfer and returns to RTE. When an interrupt occurs on the channel, indicating continuation or completion of the transfer, RTE again transfers control to the driver.

The RTE System currently includes five standard I/O drivers:

DVRØØ	- Teleprinter driver
DVRØ1	- Photo-reader driver
DVRØ2	- Paper tape punch driver
DVR12	- Line printer driver
DVR3Ø	- Disc/drum driver

The driver name consists of the letters "DVR" added to the equipment type code (see *Equipment Table*). In addition, the programmer can write drivers for special devices, following the guidelines in this section. The driver is only responsible for updating the STATUS field in the EQT entry; RTE handles the availability field.

Programmed I/O

A background or real-time program makes an EXEC call to initiate I/O transfers. RTE suspends the program until the transfer is complete so that other programs competing for execution time can have control.

PLANNING I/O DRIVERS

Before attempting to program an I/O driver, the programmer should be thoroughly familiar with Hewlett-Packard computer hardware I/O organization, interface kits, computer I/O instructions and DMA.

An I/O driver, operating under control of the Input/Output Control (IOC) and Central Interrupt Control (CIC) modules of RTE, is responsible for all data transfer between an I/O device and the computer. The device EQT entry contains the parameters of the transfer, and the Base Page Communication Area contains the number of the allocated DMA channel, if required.

An I/O Driver includes two relocatable, closed subroutines - the Initiation Section and the Completion Section. If *nn* is the octal Equipment Type Code of the device, I.*nn* and C.*nn* are the entry point names of the two sections, and DVR*nn* is the driver name.

Initiation Section

The IOC module of RTE calls the initiation section directly when an I/O transfer is initiated. Locations EQT1 through EQT11 of the Base Page Communication Area (see Appendix A) contain the addresses of the appropriate EQT entry. CHAN in base page contains the number of the DMA channel assigned to the device, if needed. This section is entered by a jump subroutine to the entry point, I.*nn*. The A-register contains the select code (channel number) of the device (bits Ø through 5 of EQT entry word 4). The driver returns to IOC by an indirect jump through I.*nn*.

Before transferring to I.*nn*, RTE places the request parameters from the user program's EXEC call into words 6 through 10 of the EQT entry. Word 6, CONWD, is modified to contain the request code in bits Ø through 5 in place of the logical unit. See the EQT entry diagram on the next page and Section IV, READ/WRITE EXEC CALL, for details of the parameters.

EQUIPMENT TABLE ENTRY DIAGRAM

WORD	CONTENTS (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)				
	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●				
1	Device Suspended List Pointer				
2	Driver "Initiation" Section Address				
3	Driver "Completion" Section Address				
4	D	B	(not used)	Unit #	Channel #
5	AV	EQUIP. TYPE CODE	STATUS		
6	CONWD (Current I/O Request Control Word)				
7	Request Buffer Address				
8	Request Buffer Length				
9	Temporary or Disc Track #				
1∅	Temporary or Starting Sector #				
11	Temporary Storage for Driver				

Figure 6-1. Equipment Table Entry Diagram

Once initiated, the driver can use words 6 through 11 of the EQT entry in any way, but words 1 through 4 must *not* be altered. The driver updates the status field in word 5, if appropriate, but the rest of word 5 must not be altered.

FUNCTIONS OF THE INITIATION SECTION

The Initiation Section of the driver operates with the interrupt system disabled (or as if it were disabled, in the case of privileged interrupt processing; see the discussion of special conditions under "Privileged Interrupt Processing").

The Initiation Section of the driver is responsible for these functions (as flow-charted in Figure 6-2):

1. Rejects the request and proceeds to step 5 if:
    - the device is inoperable,
    - the request code, or other of the parameters, is illegal.
2. Configures all I/O instructions in the driver to include the select code (and DMA channel) of the device.
3. Initializes DMA, if appropriate.

4. Initializes software flags and activates the device. All variable information pertinent to the transmission must be saved in the EQT entry because the driver may be called for another device before the first operation is complete.
5. Returns to IOC with the A-register set to indicate initiation or rejection and the cause of the reject:

   If A = Ø, then operation was initiated.

   If A ≠ Ø, then operation rejected because:

   1 - read or write illegal for device,

   2 - control request illegal or undefined,

   3 - equipment malfunction or not ready,

   4 - immediate completion (for control requests).

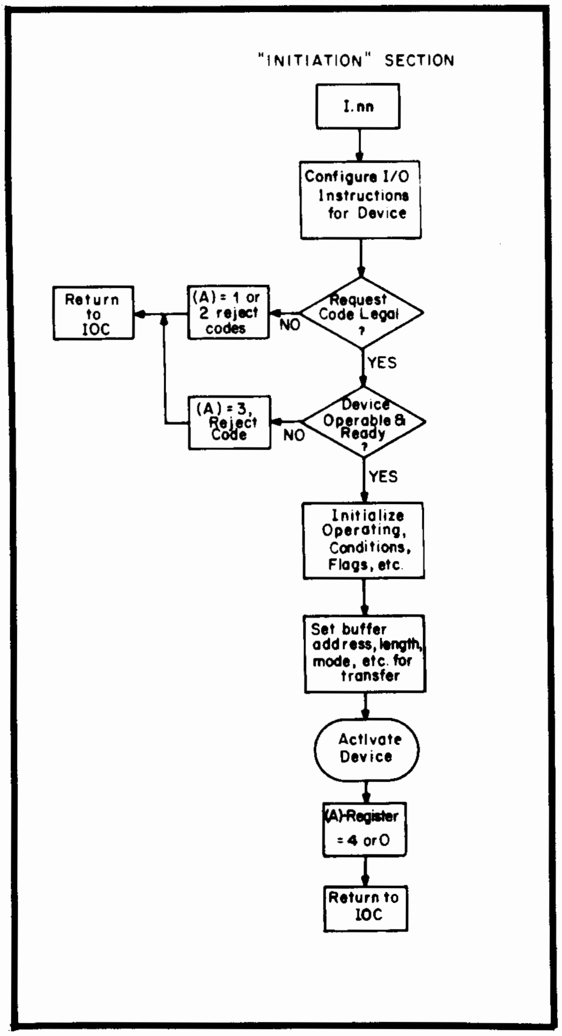
"INITIATION" SECTION
I.nn
Configure I/O Instructions for Device
Request Code Legal ?
NO
(A) = 1 or 2 reject codes
Return to IOC
YES
Device Operable & Ready ?
NO
(A) = 3, Reject Code
YES
Initialize Operating, Conditions, Flags, etc.
Set buffer address, length, mode, etc. for transfer
Activate Device
(A)-Register = 4 or 0
Return to IOC

Figure 6-2. I/O Driver Initiation Section

DMA INITIALIZATION

If a driver requires DMA, special processing must be included in the driver. After disabling the interrupt system, initiating DMA, and clearing control, the driver sets a software flag to indicate that a DMA channel is active.

The software flag is either the first or second word of the interrupt table, depending on which DMA channel is used. The flag is set by making bit 15 equal to 1.

INTBL(1) - channel 1 (location 6)
INTBL(2) - channel 2 (location 7)

The address of INTBL is contained in the word INTBA in the Base Page Communication Area. When bit 15 is set, the rest of the word *must* not be altered. The operation can be performed only if DUMMY is non-zero (meaning the system includes privileged interrupt processing.

The following code demonstrates these principles:

```
      CLF Ø                  Disable interrupts.
      STC DMA,C              Initiate DMA.
      CLA           ⎫        Bypass this section if
      CPA DUMMY     ⎬        DUMMY = Ø and special
      JMP X         ⎭        processing is not needed.
      CLC DMA       ⎫        Clear DMA control.
      LDB INTBA     ⎪        Set B = address of the
      LDA CHAN      ⎬        appropriate entry in
      CPA = D7      ⎪        the Interrupt Table.
      INB           ⎭
      LDA B,I       ⎫        Set bit 15 of the entry
      IOR = B1ØØØØØ ⎬        equal to 1 and return
      STA B,I       ⎭        to the Interrupt Table.
      STF Ø                  Enable interrupt system.
   X  continue
```

Completion Section

RTE calls the completion section of the driver whenever an interrupt is recognized on a device associated with the driver. Before calling the driver, CIC sets the EQT entry addresses in base page, sets the interrupt source code (select code) in the A-register, and clears the I/O interface or DMA flag. The interrupt system is diabled (or appears to be disabled if privileged interrupt processing is present). The calling sequence for the completion section is:

Location	Action
	Set A-register equal to interrupt source code
(P)	JSB C.*nn*
(P+1)	completion return from C.*nn*
(P+2)	continuation or error retry return from C.*nn*

The return points from C.*nn* to CIC indicate whether the transfer is continuing or has been completed (in which case, end-of-operation status is returned also).

The Completion Section of the driver is responsible for these functions (as flowcharted in Figure 6-3):

1. Checks whether word 1 (device suspended list pointer) of the EQT entry equals zero. If it does, a spurious interrupt has occurred (i.e., no I/O operation was in process on the device). The driver ignores the interrupt and makes a continuation return. If not zero, the driver configures all I/O instructions in the Completion Section to reference the interrupting device, and then proceeds to step 2.

2. If both DMA and the device completion interrupts are expected and the device interrupt is significant, the DMA interrupt is ignored by returning to CIC in a continuation return.

3. Performs the input or output of the next data item if the device is driven under program control. If the transfer is not completed, the driver proceeds to step 6.

4. If the driver detects a transmission error, it can re-initiate the transfer and attempt a retransmission. A counter for the number of retry attempts can be kept in EQT 11. The return to CIC must be (P+2) as in step 6.

5. At the end of a successful transfer or after completing the retry procedure, the following information must be set before returning to CIC at (P+1):

a. Set the actual or simulated device status, into bits Ø through 7 of EQT word 5.

b. Set the number of words or characters (depending on which the user requested) transmitted into the B-register.

c. Set the A-register to indicate successful or unsuccessful completion and the reason:
A equals Ø for successful operation,
A does not equal Ø for unsuccessful:
   1 - device malfunction or not ready,
   2 - end-of-tape (information),
   3 - transmission parity error.

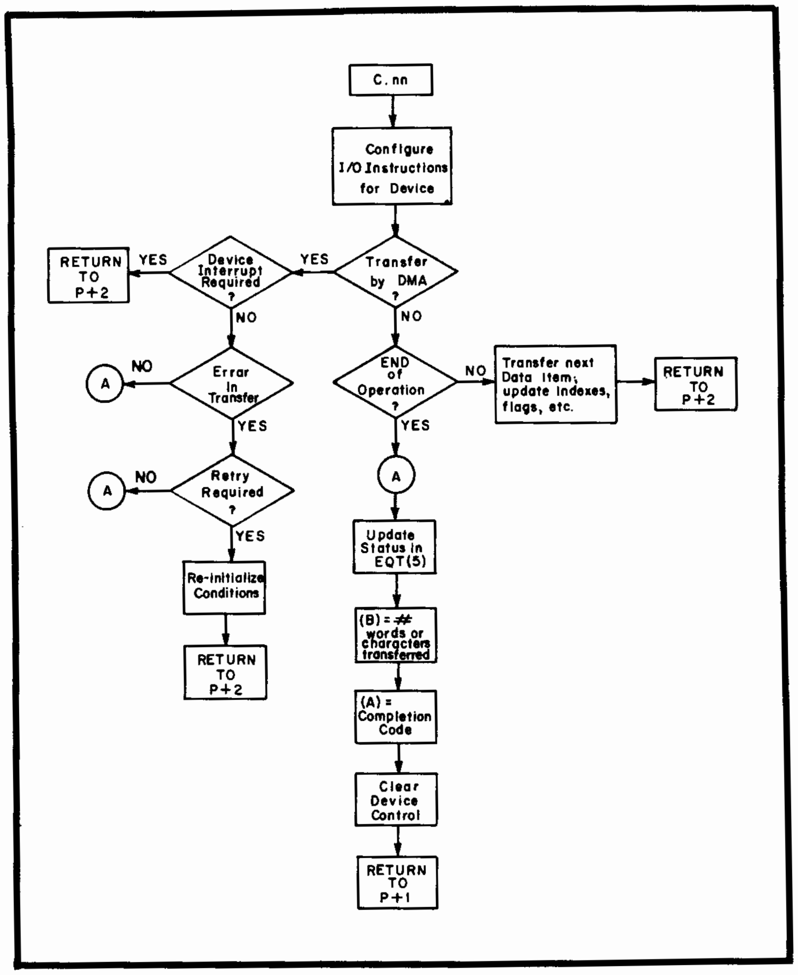

Figure 6-3. I/O Driver Completion Section

6. Clears the device and DMA control, if end-of-operation, or sets the device and DMA for the next transfer or retry. Returns to CIC at:

(P+1) - completion, with the A and B-registers set as in step 5.
(P+2) - continuation; the registers are not significant.

SAMPLE I/O DRIVER

The sample driver listed below demonstrates the principles involved in programming an I/O driver for the RTE System. Note that this driver is for tutorial purposes only and not one of the drivers supplied with the RTE System.

```
PAGE 0002

                ASMB,R,L
  00000               NAM DVR70
*
*
                      ENT I.70,C.70
*
*
*   DRIVER 70 OPERATES UNDER THE CONTROL OF THE
* I/O CONTROL MODULE OFTHE REAL-TIME EXECUTIVE.
* THIS DRIVER ID RESPONSIBLE FOR CONTROLLING
* OUTPUT TRANSMISSION TO A 16 BIT EXTERNAL
* DEVICE. <70> IS THE EQUIPMENT TYPE CODE ASSIGNED
* GENERALLY TO THESE DEVICES. I.70 IS THE
* ENTRY POINT FOR THE *INITIATION* SECTION AND C.70
* IS THE *COMPLETION SECTION ENTRY.
*
* - THE INITIATION SECTION IS CALLED FROMI/O
*   CONTROL TO INITIALIZE A DEVICE AND INITIATE
*   AN OUTPUT OPERATION.
*
*    I/O CONTROL SETS THE ADDRESS OF EACH WORD OF THE
*    11 WORD EQT ENTRY (FOR THE DEVICE)IN THE SYSTEM
*    COMMUNICATIONS AREA FOR BOTH INITIATOR AND CONTINUATOR
*    SECTIONS. THE DRIVER REFERENCES TO THE EQT ARE:
*    -EQT1 THRU EQT11-
*
*     CALLING SEQUENCE:
*
*         (A) = SELECT CODE OF THE I/O DEVICE.
*         P   JSB I.70
*         P+1 -RETURN-
*
*
*            (A) = 0, OPERATION INITIATED
*            (A) = REJECT CODE
*
*                1. ILLEGAL REQUEST
*                2. ILLEGAL MODE
*
```

```
PAGE 0003 #01  ** RT EXEC DRIVER <70> G.P.R.(OUTPUT) **


* -THE COMPLETION SECTION IS CALLED BY CENTRAL
*   INTERRUPT CONTROL TO CONTINUE OR COMPLETE
*   AN OPERATION.
*
*     CALLING SEQUENCE:
*
*        (A) = SELECT CODE OF THE I/O DEVICE.
*
*        P   JSB C.70
*        P+1 -COMPLETION RETURN-
*        P+2   CONTINUATION RETURN-
*
*          (A) = 0, SUCCESSFUL COMPLETION WITH
*                (B) = # OF WORDS TRANSMITTED.
*           (A) =2,TRANSMISSION ERROR DETECTED
*                (B), SAME AS FOR (A)=0

*
*         - CONTINUATION RETURN: REGISTERS
*             MEANING LESS
*
* -RECORD FORMAT-
*
*    THIS DRIVER PROVIDES A 16 BIT BINARY
*     WORD TRANSFER ONLY.
```

```
PAGE 0004 #01  < DRIVER 70 *INITIATION* SECTION >

*     *** INITIATION SECTION ***
*
00000 000000  I.70  NOP            ENTRY FROM IOC
*
00001 016063R       JSB SETIO      SET I/O INSTRUCTIONS FOR DEVICE
*
00002 161665        LDA EQT6,I     GET CONTROL WORD OF REQUEST,
00003 012077R       AND =B3         ISOLATE.
*
00004 052100R       CPA =B1        IF REQUEST IS FOR INPUT
00005 126000R       JMP I.70,I      THEN REJECT.
00006 052101R       CPA =B2        PROCESS FOR WRITE REQUEST
00007 026012R       JMP D.X1        GO TO WRITE REQUEST
*
* REQUEST ERROR- CAUSE REJECT RETURN TO I/O CONTROL.
*
00010 062101R       LDA =B2        SET A=2 FOR ILLEGAL CONTRL REQ.
00011 126000R       JMP I.70,I     -EXIT-
*
* WRITE REQUEST PROCESSING
*
00012 161666  D.X1  LDA EQT7,I     GET REQUEST BUFFER ADDRESS
00013 171670        STA EQT9,I      AND SET AS CURRENT ADDRESS
00014 161667        LDA EQT8,I     GET BUFFER LENGTH
00015 003004        CMA,INA         SET NEGATIVE AND SAVE
00016 171671        STA EQT10,I     AS CURRENT BUFFER LENGTH.
00017 002002        SZA            CHECK LENGTH
00020 026023R       JMP D.X3       NON-ZERO
00021 062102R       LDA =B4        IMMEDIATE COMPLETION
00022 126000R       JMP I.70,I     IF ZERO
*
* CALL COMPLETION SECTION TO WRITE FIRST WORD.
*
00023 062076R D.X3  LDA P2         ADJUST RETURN
00024 072030R       STA C.70        TO INITIATOR SECTION
00025 026032R       JMP D.X2       GO TO COMPLETION SECTION
*
00026 002400  IEXIT CLA            RETURN TO I/O CONTROL WITH
00027 126000R       JMP I.70,I      OPERATION INITIATED.
*
```

```
PAGE 0005 #01  < DRIVER 70 *COMPLETION SECTION* >

*
*     *** COMPLETION SECTION ***
*
00030 000000  C.70  NOP           ENTRY
*
00031 016063R       JSB SETIO     SET I/O INSTRUCTIONS FOR DEVICE
*
00032 002400  D.X2  CLA           IF CURRENT BUFFER LENGTH = 0,
00033 151671        CPA EQT10,I    THEN,GO TO
00034 026046R       JMP I.3        STATUS SECTION.
*
00035 165670        LDB EQT9,I    GET CURRENT BUFFER ADDRESS
00036 135670        ISZ EQT9,I    ADD 1 FOR NEXT WORD
00037 160001        LDA B,I       GET WORD
00040 135671        ISZ EQT10,I   AND INDEX WORD COUNT
00041 000000        NOP           IGNORE P+1 IF LAST WORD.
*
* OUTPUT WORD
*
00042 102600  I.1   OTA 0         OUTPUT WORD TO INTERFACE
00043 103700  I.2   STC 0,C       TURN DEVICE ON
*
00044 036030R       ISZ C.70      ADJUST RETURN TO P+2
00045 126030R       JMP C.70,I    -EXIT-
*
```

```
PAGE 0006 #01  < DRIVER 70 *COMPLETION SECTION* >

*
* STATUS AND COMPLETION SECTION.
*
  00046 102500  I.3   LIA 0          GET STATUS WORD
  00047 012103R       AND =B77       STRIP OFF BITS
  00050 070001        STA B           AND SAVE IN B
  00051 161664        LDA EQT5,I     REMOVE PREVIOUS
  00052 012104R       AND =B177400    STATUS BITS
  00053 030001        IOR B          SET NEW
  00054 171664        STA EQT5,I      STATUS BITS
*
  00055 002400        CLA            SET NORMAL RETURN COND
  00056 056102R       CPB =B4        ERROR STATUS BIT ON?
  00057 062101R       LDA =B2        YES,SET ERROR RETURN
*
  00060 165667        LDB EQT8,I     SET (B) = TRANSMISSION LOG
*
  00061 106700  I.4   CLC 0          CLEAR DEVICE
*
  00062 126030R       JMP C.70,I     -EXIT FOR COMPLETION
*
*
* SUBROUTINE <SETIO> CONFIGURES I/O INSTRUCTIONS.
*
  00063 000000  SETIO NOP
  00064 032075R       IOR LIA        COMBINE LIA WITH I/O
  00065 072046R       STA I.3        SELECT CODE AND SET.
*
  00066 042105R       ADA =B100      CONSTRUCT OTA INSTRUCTION
  00067 072042R       STA I.1
*
  00070 042106R       ADA =B1100     CONSTRUCT STC,C INSTRUCTION
  00071 072043R       STA I.2
*
  00072 032107R       IOR =B4000     CONSTRUCT CLC INSTRUCTION
  00073 072061R       STA I.4
*
  00074 126063R       JMP SETIO,I    -RETURN-
*
```

```
 PAGE 0007 #01  < DRIVER 70 *COMPLETION SECTION* >


0173*
0174* CONSTANT AND STORAGE AREA
0175*
0176  00000            A      EQU 0          A-REGISTER
0177  00001            B      EQU 1          B-REGISTER
0178*
0179  00075 102500     LIA    LIA 0
0180  00076 000025R P2     DEF IEXIT-1
0182*
0183*** SYSTEM AND BASE PAGE COMMUNICATIONS AREA ***
0184*
0185  01650                   EQU 1650B
0186*
0187*   I/O MODULE/DRIVER COMMUNICATION
0188*
0189  01660            EQT1   EQU .+8
0190  01661            EQT2   EQU .+9
0191  01662            EQT3   EQU .+10
0192  01663            EQT4   EQU .+11
0193  01664            EQT5   EQU .+12
0194  01665            EQT6   EQU .+13
0195  01666            EQT7   EQU .+14
0196  01667            EQT8   EQU .+15
0197  01670            EQT9   EQU .+16
0198  01671            EQT10  EQU .+17
0199  01672            EQT11  EQU .+18
0200*
0201*
      00077 000003
      00100 000001
      00101 000002
      00102 000004
      00103 000077
      00104 177400
      00105 000100
      00106 001100
      00107 004000
0202                          END
**  NO ERRORS*
```

PRIVILEGED INTERRUPT PROCESSING

When a special I/O interface card, number 12620A, is included in the system, RTE allows a class of privileged interrupts to be processed independently of regular RTE operation, with a minimal delay in responding to interrupts.

The special I/O card separates the privileged (high priority, low-channel numbers) interrupts from the regular system-controlled interrupts. When this card is present, RTE does not operate with the interrupt system disabled, but rather, sets control on the special I/O card to hold off lower-priority interrupts. The privileged interrupts are enabled when RTE is running and they can interrupt any RTE operation.

The routines which handle privileged interrupts must be completely independent of RTE.

Selection of the Privileged Interrupt Option

The presence and location of the special I/O card, which signifies privileged interrupt processing, is selected at system generation time by RTGEN. Its actual hardware location is stored in the word DUMMY in base page (or if not available, zero). See Section VII for the exact specification procedure.

Privileged Interrupts

The privileged interrupts can be recognized within 100 microseconds and processed by special routines embedded in the system area for high-speed program controlled data transmission.

A "JSB --,I" in the interrupt location (set by using "ENT,*name*" when configuring the interrupt table) channels the special interrupt directly to the entry point of its associated special routine. CIC and the rest of RTE are not aware of these interrupts.For the benefit of privileged interrupts, CIC sets a software flag indicating the status of the memory protect facility, MPTFL, in base page.

If MPTFL equals zero, the memory protect is "ON". Any special interrupt routine must issue a "STC 5" instruction immediately before returning to the point of interruption by a "JMP ---,I".

If MPTFL equals one, RTE itself was executing when the privileged interrupt occurred, and memory protect is "OFF". The special routine must not issue the "STC 5" in this case.

Special Processing by CIC

During interrupt processing, CIC saves the registers, issues a CLF instruction to the interrupt location, and checks the DUMMY flag. If the flag is zero, the hardware interrupt system is left disabled and normal processing continues. If non-zero, the value is used to issue a STC to the I/O location (this assumes that the flag on the special I/O card is set); the STC holds off lower priority interrupts until the control is cleared on the special card.

CIC sets the memory protect flag, MPTFL, equal to one, clears the control flip-flop on each DMA channel to defer DMA completion interrupts, enables the interrupt system (a zero is in the interrupt location for the special card so that interrupts from the card are ignored.)

$IRT, a special subroutine within CIC, resets the flags and DMA channels upon completion of normal system processing. Other modules of RTE use this routine also. $IRT also sets MPTFL = Ø to indicate that memory protect is "ON", and sets the control flip-flop on the active DMA channels,if bit 15 of the DMA interrupt entries equals one.

Privileged Interrupt Routines

A privileged interrupt routine must save and restore all registers and restore memory protect to its original state prior to special interrupt (word MPTFL contains this status) because any interrupt automatically disables memory protect.

A privileged interrupt routine must *not* use any features or requests of RTE, nor use either DMA channel. It can communicate with normal user programs by use of the appropriate COMMON region. Flags, parameters, control words, etc., can be set and monitored by either routine in pre-defined locations in COMMON. The starting address of either COMMON region is available in Base Page. (See Appendix A.) A normal user program can be scheduled to run at periodic time intervals to scan and set indicators in COMMON.

SECTION VII
RTE SYSTEM INSTALLATION

The set up and operation of a Real-Time Executive System involves two essential steps and one optional step: the RTE System must be configured using the Real-Time System Generator, RTGEN; it must be initiated from the disc by the core-resident Basic Binary Disc Loader, BBDL; and it may be dumped onto tape using the System Dump, SDUMP, as protection against a disc failure.

This section describes the three routines, RTGEN, BBDL, and SDUMP, that are responsible for these processes.

RTGEN, THE REAL-TIME SYSTEM GENERATOR

RTGEN configures a RTE System to fit a particular user's core memory size, I/O equipment, and programming needs.

To accomplish this, RTGEN requests certain information from the user; then it accepts the relocatable program modules to be included in the system, determines where they belong in core, relocates them into absolute format, and stores them on the disc. RTGEN also creates I/O tables by identifying each I/O device and its associated driver routine, and establishing procedures for interrupt processing on each channel.

RTGEN is an absolute program, loaded into core by the Basic Binary Disc Loader from paper tape. Since RTGEN is independent of the RTE System which it generates, the I/O operations of RTGEN require special programs called SIO Drivers.

Using other standard Hewlett-Packard Software, the user can create a magnetic tape or disc file of the relocatable program modules for quick and easy configuration.

RTGEN operates on the same minimum configuration as that required for a RTE System.

Operating Procedures

The operation of RTGEN involves four phases:

▯ INITIALIZATION PHASE. RTGEN requests specifications for the RTE System, including description of available disc space, memory, time base generation channel, swapping option, and program input devices.

▯ PROGRAM INPUT PHASE. The operator inputs the relocatable programs provided with the system and created by the user.

▯ PARAMETER INPUT PHASE. Parameters describing or changing the type, priority, and execution interval of each program may be entered. (Although this information may already be included in the program's NAM record, it can be changed at this point.) RTGEN requests the number of blank ID segments to be reserved for subsequent program addition.

▯ DISC LOADING PHASE. RTGEN requests a specification of the base page linkage, and begins loading programs onto the disc. Systems programs (i.e., the modules of RTE), are loaded first, after which RTGEN then requests information for the Equipment Table, Device Reference Table, and Interrupt Table and proceeds to load the rest of the programs onto the disc.

To execute RTGEN and configure a RTE System, follow these steps:

▯ Turn on all equipment; set the system teleprinter to LINE, and disable the disc protect. (See Drum Memory Interface Kit Manual 12610-9001, or Disc Memory Interface Kit Manual 12606-9001.)

▯ Load the RTGEN tape into core using BBDL, the core-resident loader, and add the appropriate SIO Drivers. (If the relocatable programs are on a magnetic tape or disc file, the file must be created by the Prepare Tape System (PTS) and the magnetic tape or disc SIO Driver must be added): Refer to the *PREPARE TAPE SYSTEM* reference manual for a description of PTS.

a. Load the SIO Buffered Teleprinter Driver tape using the BBDL.
b. Set the switch register to 2_8 and press LOAD ADDRESS.
c. Set switch register bits 5-0 to the channel number of the device associated with the driver.
d. Press RUN.
e. Repeat these steps, if appropriate, for the Punched Tape Reader and the Magnetic Tape Unit or Disc driver.

▯ Set the switch register to $1ØØ_8$, press LOAD ADDRESS, then press RUN. RTGEN begins the initialization phase.

INITIALIZATION PHASE

During the initialization phase, RTGEN requests information necessary to begin generating the RTE System. After each question prints, the operator responds with the required answer. The following dialogue is typical (operator responses are only examples; actual responses should be appropriate to the particular system being generated).

RTGEN requests the octal channel number (higher priority channel if multi-card) of the system disc or drum unit.....................................DISC CHNL?

Operator responds............................ *2Ø*

RTGEN requests the number of tracks (decimal) on the system disc or drum. (If a relocatable file exists on the unit, subtract these tracks from the size.)..SYS DISC SIZE?

Operator responds............................ *32*

RTGEN requests the number of hardware protected tracks...NO. PROTECTED?

Operator responds............................ *16*

RTGEN requests the number of sectors (decimal) per track on the system disc or drum. (Disc units are usually 90; drums, 128.)....................................#SECTORS/TRACK?

Operator responds.................................... 9Ø

RTGEN requests the number of tracks on the auxiliary disc or drum..AUX DISC SIZE?

Operator responds with number of tracks (or zero (Ø), if there is no auxiliary disc).............. 22

If the response to the last query was not zero, RTGEN requests the number of sectors (decimal) per track on the system disc or drum.........................#SECTORS/TRACK?

Operator responds.................................... 9Ø

RTGEN requests the I/O channel of the Time Base Generator (octal)...TBG CHNL?

Operator responds.................................... 1Ø

RTGEN requests the address of the privileged interrupt I/O card (if present)...............................PRIV.INT.CARD ADDR?

Operator responds with the octal channel number of the privileged interrupt card (and all devices in channels below the card become privileged) or zero (Ø) if the card is not used................................ Ø

RTGEN asks whether the swapping option (which allows real-time disc-resident programs to swap in and out of core according to priority) is to be included in the RTE System...SWAPPING?

Operator responds with YES or NO..................... *YES or NO*

RTGEN requests the last word of available core memory, in octal...LWA MEM?

Operator responds.................................. *37677*

RTGEN requests the type of input unit for relocatable program modules...PRGM INPT?

Operator responds with PT (for paper tape), TY (for teleprinter), MT (for magnetic tape), or DF (for disc file)...................................... *PT*

RTGEN requests the type of input unit for relocatable library programs...LIBR INPT?

Operator responds with PT, TY, MT, or DF............. *MT*

RTGEN requests the type of input unit for parameters, describing the relocatable programs.........................PRAM INPT?

Operator responds with PT or TY...................... *TY*

When RTGEN finishes the initialization phase, the computer halts. The operator must press RUN to continue to the next phase.

PROGRAM INPUT PHASE

During the program input phase, RTGEN accepts relocatable programs from the Program Input Unit and Library Input Unit specified during the initialization phase. The operator selects the input device by setting switch register bits Ø-1 ($ØØ_2$ for the Program Input Unit, or $1Ø_2$ for the Library Input Unit), and places the programs in the input device. Background main programs must enter prior to their segments. The RTE/DOS FORTRAN IV Library or the RTE/DOS Basic Formatter should be loaded before the RTE/DOS Relocatable Library.

If a program was not generated with the type code in the NAM record, the operator must set switch register bits 3-5 to the type code (Ø to 7). However, programs on a magnetic tape or disc file must have the type code in the NAM record.

The operator presses RUN. When entering paper tape, the message "*EOT" is printed whenever an end-of-tape occurs. The computer halts.

At this point, the operator has several options: additional programs can be input from the same device by repeating the steps above; input can be switched to the other input device (by setting the switch register bits to $ØØ_2$ or $1Ø_2$). If there are no undefined externals, this message is printed: NO UNDEF EXTS

To terminate the program input phase, set switch register bits Ø-1 to $Ø1_2$, and press RUN twice (the first RUN prints any undefined external references).

PARAMETER INPUT PHASE

During the parameter input phase, the operator can modify the type, priority, or execution intervals of any of the programs entered during the Program Input Phase (except that main programs and segments cannot be changed).

Each parameter record is of this general form:

name, type [,*priority*][,*execution interval*] where *name* is the name of the program,

type is the program type code:

- 1 - Real-Time resident
- 2 - Real-Time disc-resident
- 3 - Background disc-resident
- 4 - Background resident
- 5 - Background segment
- 6 - Re-entrant or privileged library
- 7 - Utility
- >7 - Program deleted from the system

priority is the program priority from 1 to 99, with 1 the highest,

execution interval is a list of six parameters specifying the times the program should be scheduled for execution, once it is turned on. The first two values specify the execution interval, and the last four specify an initial offset starting time:

resolution code (Ø to 4):

- Ø - no execution interval
- 1 - tens of milliseconds
- 2 - seconds
- 3 - minutes
- 4 - hours

execution multiple (Ø to 999); the *resolution code* gives the units for the *execution multiple.*

initial offset (four values):

- *hours* (Ø-23)
- *minutes* (Ø-59)
- *seconds* (Ø-59)
- *tens of milliseconds* (Ø-99)

To end the parameter input phase and continue on to the disc loading phase, the operator enters "/E" instead of a parameter record. RTGEN then asks two questions before entering the disc loading phase:

RTGEN requests the number of blank ID segments to be allocated for on-line loading of background programs by the Relocating Loader.................................#OF BLANK ID SEGMENTS?

Operator responds with a one or two digit decimal number (zero is changed to one, because one is required to do any on-line loading); 28 words are reserved in the resident table area for each blank ID segment... 1∅

DISC LOADING PHASE

RTGEN requests the first word of available core memory in Base Page..FWA BP LINKAGE?

Operator responds with an octal number just above the locations necessary for interrupt linkages (see below)... 4∅

Disc loading begins with the modules of the Real-Time Executive, including I/O Drivers. As RTGEN loads these programs, it prints a memory map giving the starting locations and, if switch register bit 15 is up, the entry points for all main programs and subroutines (subroutines are indented two spaces).

Next, RTGEN generates the three I/O tables: Equipment Table, Device Reference Table, and the Interrupt Table.

RTGEN requests the Equipment Table entries..................EQUIPMENT TABLE ENTRY

Operator responds with a series of one line EQT entries, which are assigned EQT numbers sequentially from one as they are entered. The EQT entry relates the EQT number to an I/O channel and driver, in this format:.............................. *nl*,*DVRnn*,[,*D*][,*B*][,*U*]

where *nl* is the I/O channel (lower number if multi-board),

DVRnn is the driver name(*nn* is the Equipment Type Code, see Section VI EQUIPMENT TABLE),

D, if present, means DMA channel required,

B if present, means automatic output buffering required (line printer cannot be buffered),

U, is the physical sub-channel number.

Operator terminates the Equipment Table entries by typing.. /E

NOTE: The system disc should be the first EQT entry to permit top priority assignment to an available DMA channel.

RTGEN requests the logical unit assignments for the Device Reference Table......................................DEVICE REFERENCE TABLE

For each logical unit number, RTGEN prints................. *n*=EQT#?
where *n* is a decimal integer starting with one.

Operator responds with an EQT entry number appropriate to the standard definition of *n*. (See Section VI, LOGICAL UNIT NUMBERS.) Numbers above 7 may be assigned any EQT entry disired.. *4*

Operator terminates entry by typing................. /E

RTGEN requests the Interrupt Table entries..................INTERRUPT TABLE

Operator responds with an entry for each I/O location which may interrupt, in ascending order, and in this format........................... *nl, option*

where *n1* is the octal channel number (must be entered in ascending order), and
option directs RTE in handling the interrupt; there are four options:
EQT,*n2* - relates channel to EQT entry *n2*.
PRG,*name* - causes program *name* to be scheduled upon interrupt.
ENT,*entry* - causes control to transfer to the *entry* point of a user-written system program upon interrupt.
ABS,*xxxxxx* - places an absolute octal value in the interrupt location (may be NOP, CLC, etc.).

Operator terminates entry by typing.................. /E

Following loading of the resident library, real-time resident and disc-resident programs, RTGEN reports the first word address of the system available memory...........FWA SY MEM *nnnnn*

Next, RTGEN requests any changes to this address (in order to increase the real-time disc-resident area for future addition of larger programs). At least 150_{10} words of base page linkage area are reserved for the real-time disc resident area................CHANGE FWA SY AV MEM?

Operator responds with zero for no change, or an octal address greater than *nnnnn* (ERR 14 printed if new is less than old)...................... *xxxxx*

Finally, RTGEN requests the first address of the Background Area (the area between the FWA SY MEM and the background is used for temporary storage of buffers, re-entrant temporaries, etc; it should be large enough to handle the largest anticipated I/O transfer)...BG BOUNDARY?

Operator responds with an octal address.............. *xxxxx*

RTGEN proceeds to load the background resident and disc-resident programs, and prints..........................SYSTEM STORED ON DISC

RTGEN reports the address of the last sector and track used in storing the RTE System on the disc, and halts...LAST SYS DISC ADDR: TRK *nn* SEC *mmm* (8)

where both *nn*, the track number, and *mmm*, the sector number, are octal numbers. These should be recorded if the RTE System is going to be dumped onto tape for back-up protection using SDUMP. See Appendix C for the listing of a sample RTGEN execution.

Restart

During any of the phases, RTGEN can restart that phase if any error occurs. The operator sets the switch register equal to $1ØØ_8$, and presses LOAD ADDRESS and RUN.

In addition, the parameter input phase can be re-entered at $4ØØØ_8$, and the disc loading phase at $6ØØØ_8$.

Error Messages

The following messages may be printed on the teleprinter during execution of RTGEN:

Message	Meaning	Action
Messages During Initialization and Input Phase		
ERR Ø1	Invalid response to initialization request.	Message is repeated. Enter valid reply.

Message	Meaning	Action
ERR Ø2	Checksum error on program input.	Computer halts; reposition tape to beginning of record and press RUN to reread.
ERR Ø3	Record out of sequence.	Same as ERR Ø2.
ERR Ø4	Illegal record type.	Same as ERR Ø2.
ERR Ø5 *name*	Duplicate entry point.	Revise program by re-labeling the entry points (the current entry point replaces the previous entry point).
ERR Ø6	Invalid base page length (must be zero).	Base page area is ignored, but memory protect error will occur if program is executed.
ERR Ø7	Program Name or Entry Point Table overflow of available memory.	Irrecoverable error. Revise or delete programs.
ERR Ø8 *name*	Duplicate program name.	The current program replaces the previous program.

Messages During the Parameter Phase

ERR Ø9	Parameter name error (no such program).	Enter valid parameter statement.
ERR 1Ø	Parameter type error.	Same as ERR Ø9.
ERR 11	Parameter priority error.	Same as ERR Ø9.
ERR 12	Execution interval error.	Same as ERR Ø9.

General Messages

ERR 13	BG segment precedes BG main disc-resident program.	Irrecoverable.
ERR 14	Invalid background bounds or illegal response to CHANGE FWA SYS MEM?	Message is repeated. Enter valid reply.
ERR 15	More than 63 subprograms called by a main program.	Revise main program (subsequent calls to subprograms are ignored).

Message	Meaning	Action
ERR 16	Base page linkage overflow into system communication area.	Diagnostic printed for each word required (communication area is used). Revise order and composition of program loading to reduce linkage requirements.
ERR 17	Current disc address exceeds number of available tracks.	Irrecoverable error.
ERR 18	Memory overflow (absolute code exceeds LWA memory).	Diagnostic printed for each word required (absolute code is generated beyond LWA). Revise program.
ERR 19	Program overlay (current word of absolute code has identical location to previous).	Current word (the address is printed) is ignored.
Err 2Ø	Binary DBL record overflow of internal table.	Records overlay previous DBL records (diagnostic printed for each overflow record). Revise program.
ERR 21	Module containing entry point $CIC not loaded.	Irrecoverable error.
ERR 22	Read parity/decode disc error. A-register bits 7-14 show track number; bits 0-6 show sector number.	After ten attempts to read or write the disc sector, the computer halts. To try ten more times, press RUN.
ERR 23	Invalid FWA BP LINKAGE.	Message repeated; enter valid reply.

Messages During I/O Table Entry

ERR 24	Invalid channel number.	Enter valid EQT statement.
ERR 25	Invalid driver name or no driver entry points.	Same as ERR 24.
ERR 26	Invalid or duplicate D,B,U operands.	Same as ERR 24.

Message	Meaning	Action
ERR 27	Invalid logical unit no.	Enter valid DRT statement.
ERR 28	Invalid channel number.	Enter valid INT statement.
ERR 29	Channel number decreasing.	Same as ERR 28.
ERR 30	Invalid mnemonic.	Same as ERR 28.
ERR 31	Invalid EQT number.	Same as ERR 28.
ERR 32	Invalid program name.	Same as ERR 28.
ERR 33	Invalid entry point.	Same as ERR 28.
ERR 34	Invalid absolute value.	Same as ERR 28.
ERR 35	Base page interrupt locations overflow into linkage area.	Re-start Disc Loading Phase at FWA BP LINKAGE? request.
ERR 36	Invalid number of characters in final operand.	Same as ERR 28.

General Message

ERR 37 *name*	Invalid declaration of common in system or library programs (*name* is the illegal program).	Revise the program.

INITIATING OPERATION FROM THE DISC

The Basic Binary Disc Loader (BBDL), a modified version of the standard Basic Binary Loader, resides in the highest, protected 64 words of core and loads either absolute format paper tapes or disc-based systems, such as RTE System.

Loading the RTE System

▯ The operator sets the switch register equal to Ø7776Ø$_8$, and presses LOAD ADDRESS. He then sets the loader switch to ENABLED, and presses RUN.

▯ When the computer halts with 1∅2∅77$_8$ in the T-register, the operator sets the loader switch to PROTECTED, sets switch register bit ∅ to ∅, and presses RUN. When RTE is loaded, it will type:

*INIT REAL TIME CLOCK

The operator either initializes the clock using the TM operator request, or types any other request.

If a halt with 1∅2∅11$_8$ in the T-register means that a checksum error occurred. 1∅2∅55$_8$ means an illegal address. If the BBDL itself is destroyed, it can be replaced through the switch register using the octal listing in Appendix A.

Paper Tape Loading

The operator places the paper tape in the teleprinter reader (or photoreader, if available). He sets the switch register equal to ∅777∅∅$_8$, and presses LOAD ADDRESS. He then sets the loader switch to ENABLED and presses RUN. After BBDL reads the tape, the operator sets the loader switch to PROTECTED.

CREATING A BACK-UP COPY

SDUMP, the System Dump, is an independent utility program that can create back-up copies of disc-based systems on punched tape or magnetic tape. The back-up copy can later be reloaded onto the disc by SDUMP.

Because it is an independent program like RTGEN, SDUMP requires independent I/O drivers, the SIO Drivers. For paper tape storage, the SIO Teleprinter Driver, SIO Paper Tape Reader Driver, and the SIO Paper Tape Punch Driver are required. For magnetic tape storage, the SIO Teleprinter Driver and SIO Magnetic Tape Unit Driver are required. The operator loads the SDUMP tape and SIO Driver tapes as described for RTGEN. The magnetic tape driver must be loaded after SDUMP.

After loading SDUMP, execution begins at $1ØØ_8$. SDUMP prints out a request guide on the teleprinter:

```
DUMP = D,T[-S][,T[-S]] ([] = OPTIONAL)
VERIFY = V
LOAD  = L
TERMINATE = T
```

Then, SDUMP requests the lower-number channel of the disc in octal.......................................DISC CHNL?

After SDUMP types COMMAND, the operator replies with V,L,T, or D (followed by parameters)...............L

D for dumping requires a parameter specifying the first track to be dumped. The end parameter is optional. The values are octal. The last track is reported by RTGEN after creating the RTE System and should be recorded then. If output is to paper tape, trailer and leader blank tape is produced, and two tracks are dumped at a time. If output is to magnetic tape, the entire information is dumped, followed by an End-Of-File which is written over by subsequent dumps.

V for verifying, involves placing the dump in the input device, reading it in, and checking each record against the contents of the disc. Comparison errors are reported. If magnetic tape is verified, only one file is checked.

L for loading causes the dumped information to be loaded back onto the disc. The information is verified after it is output to the disc, and comparison errors are reported.

An illegal command causes the message:

```
STATEMENT ERROR
```

An error in specifying the disc channel, causes the message:

```
PARAM ERROR:  NON-NUMERIC OR NON-OCTAL
```

If the magnetic tape is used, a rewind is issued during initialization, before and after a verify or load operation, and rewind/standby after T for termination.

Error Messages

The following messages may be printed on the teleprinter by SDUMP:

Statement	Action
STATEMENT ERROR	Re-type input statement in correct format.
EOT	The end of the input tape being read has been reached; either load the next tape or go on to the next phase.
CHANGE OUTPUT TAPE, HIT RUN	Two full tracks have been dumped onto paper tape; perform the requested action.
TURN OFF DISC PROTECT, HIT RUN	Set the Disc Track Protect Switch off, then press RUN.
DISC INPUT ERROR	Disc Error Diagnostic, for a Parity or Decode or Abort status after 10 re-trys. Input sequence re-peated on restart.
DISC WRITE ABORT	Disc Error Diagnostic, for an Abort status after a write attempt. Sequence is repeated if restarted.
TRACK 27(8)SECTOR 1Ø7(8)	Identification information for the Disc and Tape Error Diagnostics are described as follows:
TAPE/DISC VERIFY ERROR	Disc and tape records do not agree. Disc record is rewritten on restart.
TAPE CHECKSUM ERROR	The checksum in the tape record does not match the sum computed by SDUMP. Current record is ignored if re-started.

Continued on the next page.

Statement	Action
MT ERROR - READ PARITY MT ERROR - EOT,RESTART	Magnetic Tape Errors. Error recovery procedures are completed by driver. Restart to re-try sequence.

APPENDIX A
TABLES

This appendix contains several useful tables and diagrams.

BASE PAGE COMMUNICATION AREA

A block of storage in base page, starting at 1650_8,contains the system communication area and is used by RTE to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, etc. The Real-Time Assembler allows absolute references into this area (i.e., less than 2000_8) within relocatable programs, so that user programs can read information from this area, but cannot alter it because of the memory protect feature.

The Base Page Communication Area contains:

Octal Location	Contents	Description
SYSTEM TABLE DEFINITION		
Ø165Ø	EQTA	FWA OF EQUIPMENT TABLE
Ø1651	EQT#	NO. OF EQT ENTRIES
Ø1652	DRT	FWA OF DEVICE REFERENCE TABLE
Ø1653	LUMAX	NO. OF LOGICAL UNITS (IN DRT)
Ø1654	INTBA	FWA OF INTERRUPT TABLE
Ø1655	INTLG	NO. OF INTERRUPT TABLE ENTRIES
Ø1656	TAT	FWA OF TRACK ASSIGNMENT TABLE
Ø1657	KEYWD	FWA OF KEYWORD BLOCK
I/O MODULE/DRIVER COMMUNICATION		
Ø166Ø	EQT1	ADDRESSES
Ø1661	EQT2	
Ø1662	EQT3	OF
Ø1663	EQT4	

Octal Location	Contents	Description
Ø1664	EQT5	CURRENT
Ø1665	EQT6	
Ø1666	EQT7	11-WORD
Ø1667	EQT8	
Ø167Ø	EQT9	EQT
Ø1671	EQT1Ø	
Ø1672	EQT11	ENTRY
Ø1673	CHAN	CURRENT DMA CHANNEL NO.
Ø1674	TBG	I/O ADDRESS OF TIME-BASE CARD
Ø1675	SYSTY	EQT ENTRY ADDRESS OF SYSTEM TTY

SYSTEM REQUEST PROCESSOR/'EXEC' COMMUNICATION

Ø1676	RQCNT	NO. OF REQUEST PARAMETERS -1
Ø1677	RQRTN	RETURN POINT ADDRESS
Ø17ØØ	RQP1	
Ø17Ø1	RQP2	ADDRESSES OF REQUEST
Ø17Ø2	RQP3	
Ø17Ø3	RQP4	PARAMETERS
Ø17Ø4	RQP5	
Ø17Ø5	RQP6	(SET FOR MAXIMUM OF 8
Ø17Ø6	RQP7	PARAMETERS)
Ø17Ø7	RQP8	

ADDRESSES OF SYSTEM LISTS

Ø171Ø	DORMT	ADDRESS OF 'DORMANT' LIST
Ø1711	SKEDD	'SCHEDULE' LIST
Ø1712	SUSP1	'I/O SUSPEND' LIST
Ø1713	SUSP2	NOT USED
Ø1714	SUSP3	'AVAILABLE MEMORY' LIST
Ø1715	SUSP4	'DISC ALLOCATION' LIST
Ø1716	SUSP5	'OPERATOR SUSPEND' LIST

Octal Location	Contents	Description
DEFINITION OF EXECUTING PROGRAM ID SEGMENT'		
Ø1717	XEQT	ID SEGMENT ADDR. OF CURRENT PROG.
Ø172Ø	XLINK	'LINKAGE'
Ø1721	XTEMP	'TEMPORARY (5-WORDS)
Ø1726	XPRIO	'PRIORITY' WORD
Ø1727	XPENT	'PRIMARY ENTRY POINT'
Ø173Ø	XSUSP	'POINT OF SUSPENSION'
Ø1731	XA	'A REGISTER' AT SUSPENSION
Ø1732	XB	'B REGISTER' AT SUSPENSION
Ø1733	XEO	'E AND OVERFLOW' AT SUSPENSION
SYSTEM MODULE COMMUNICATION FLAGS		
Ø1734	OPATN	OPERATOR/KEYBOARD ATTENTION FLAG
Ø1735	OPFLG	OPERATOR COMMUNICATION FLAG
Ø1736	SWAP	RT DISC RESIDENT SWAPPING FLAG
Ø1737	DUMMY	I/O ADDRESS OF DUMMY INT. CARD
Ø174Ø	IDSDA	DISC ADDR. OF FIRST ID SEGMENT
Ø1741	IDSDP	-POSITION WITHIN SECTOR
DEFINITION OF MEMORY ALLOCATION BASES		
Ø1742	BPA1	FWA R/T DISC RES. BP LINK AREA
Ø1743	BPA2	LWA R/T DISC RES. BP LINK AREA
Ø1744	BPA3	FWA BKG DISC RES. BP LINK AREA
Ø1745	LBORG	FWA OF RESIDENT LIBRARY AREA
Ø1746	RTORG	FWA OF REAL-TIME AREA
Ø1747	RTCOM	LENGTH OF REAL-TIME COMMON AREA
Ø175Ø	RTDRA	FWA OF R/T DISC RESIDENT AREA
Ø1751	AVMEM	FWA OF SYSTEM AVAILABLE MEMORY
Ø1752	BKGRG	FWA OF BACKGROUND AREA
Ø1753	BKCOM	LENGTH OF BACKGROUND COMMON AREA
Ø1754	BKDRA	FWA OF BKG DISC RESIDENT AREA

Octal Location	Contents	Description
UTILITY PARAMETERS		
Ø1755	TATLG	LENGTH OF TRACK ASSIGNMENT TABLE
Ø1756	TATSD	# OF TRACKS ON SYSTEM DISC
Ø1757	SECT2	# SECTORS/TRACK ON LU 2 (SYSTEM)
Ø176Ø	SECT3	# SECTORS/TRACK ON LU 3 (AUX.)
Ø1761	DSCLB	DISC ADDR OF RES LIB ENTRY PTS
Ø1762	DSCLN	# OF RES LIB ENTRY POINTS
Ø1763	DSCUT	DISC ADDR OF RELOC UTILITY PROGS
Ø1764	DSCUN	# OF RELOC UTILITY PROGS
Ø1765	LGOTK	LOAD-N-GO: LU,STG TRACK,# OF TRKS
Ø1766	LGOC	CURRENT LGO TRACK/SECTOR ADDRESS
Ø1767	SFCUN	SOURCE FILE LU AND DISC ADDRESS
Ø177Ø	MPTFL	MEMORY PROTECT ON/OFF FLAG (Ø/1)
Ø1777	BKLWA	LWA OF MEMORY IN BACKGROUND

DISC LAYOUT OF RTE SYSTEM

Figure A-1 on the next page diagrams the allocation of disc space by RTGEN when it creates a RTE System.

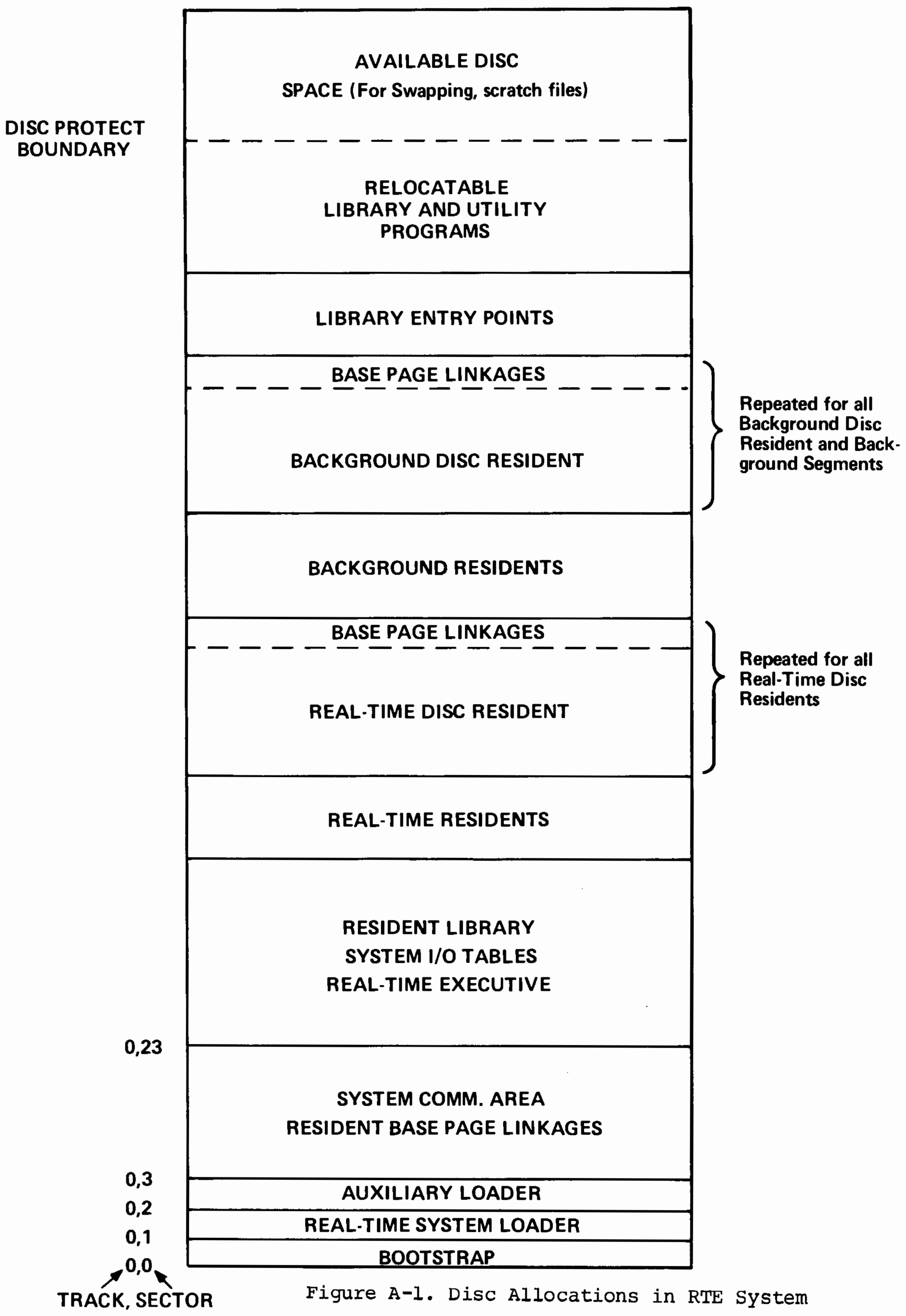

Figure A-1. Disc Allocations in RTE System

PROGRAM ID SEGMENT

Each main user program has a 22- to 28-word ID segment in the resident system area. Certain information in an ID segment is static and is set by RTGEN or the loader. Other information is variable and is maintained by the Executive. In the table below are recorded the names of information stored in the ID segment, whether it is Dynamic(d) or Static(s), and words used.

TABLE A-1
ID SEGMENTS

WORD*	LABEL	SIZE	S or D	USE
1	XLINK	1	d	List linkage
2-6	XTEMP	5	d	Temporary storage
7	XPRIO	1	s(d on-line)	Priority
8	XPENT	1	s	Primary entry point
9	XSUSP	1	d	Point of suspension
10	XA	1	d	A-Register
11	XB	1	d	B-Register
12	XEO	1	d	E- and O-Registers
13-15	NAME	3	s	Program name and type
16	STAT	1	d	Status
17	TLINK	1	d	Time linkage
18	RESL	1	d	Resolution code
19	TMSEC	1	d	Tens of milliseconds
20	TSEC	1	d	Seconds
21	TMIN	1	d	Minutes
22	THOUR	1	d	Hours

Continued on the next page.

*Words 1-12 are loaded into base page XEQT Area during execution of a program.

WORD	LABEL	SIZE	S or D	USE
For Disc Resident Programs Only:				
23-26	MEM	4	s	Mem(1) = Low main; M(3) = Low base; M(2) = Hi main; M(4) = Hi base.
27	DMAN	1	s	Source disc address (L.U./tr/sec)
28	SMAN	1	d	Swap location

BBDL LISTING

Figure A-2 is an octal listing of the Basic Binary Disc Loader that resides in the protected, highest 64 words of core. If the loader is destroyed in core, it can be replaced through the switch register using this listing. The operator simply replaces symbolic items with the value appropriate to the configuration.

	B							
A	0	1	2	3	4	5	6	7
0*m*7700:	107700	002401	063726	006700	017742	007306	027713	002006
0*m*7710:	027703	102077	027700	077754	017742	017742	074000	077757
0*m*7720:	067757	047755	002040	027740	017742	040001	177757	037757
0*m*7730:	000040	037754	027720	017742	054000	027702	102011	027700
0*m*7740:	102055	027700	000000	006600	1037*cc*	1023*cc*	027745	1064*cc*
0*m*7750:	002041	127742	005767	027744	000000	1*z*0100	0200*nn*	000000
0*m*7760:	107700	063756	102606	002700	1026*qq*	001500	102602	063777
0*m*7770:	102702	102602	103706	1027*nn*	067776	074077	024077	177700

Legend: A + B = Memory Address

m = 1 for 8K, 3 for 16K, 5 for 24K, 7 for 32K memory

nn = first disc channel

qq = second disc channel

cc = photoreader or Teleprinter address

z = 6 for 8K, 4 for 16K, 2 for 24K, Ø for 32K memory

Figure A-2. BBDL LISTING

'NAM' RECORD USED IN RTE/DOS ALGOL

PBUF	WORD COUNT = 21_8	-Ø-	
+1	ØØ1Ø ←——————→ Ø		RIC = 1
+2	CHECKSUM		
+3	P	N	5 Character Program Name
+4	A	M	
+5	E	blank	
+6	PROGRAM LENGTH		
+7	Ø ←——————→ Ø		
+8	'COMMON' LENGTH		
+9	TYPE		
+10	PRIORITY		
+11	RESOLUTION CODE		
+12	EXECUTION MULTIPLE		
+13	HOURS		
+14	MINUTES		
+15	SECONDS		
+16	TENS OF MILLISECONDS		

APPENDIX B
RELATION TO OTHER SOFTWARE

The Hewlett-Packard 2116B is a general-purpose computer; as such, it can handle standard HP software when the Real-Time Executive System is inactive. Every computer is shipped with the standard software and documentation appropriate to the system configuration.

In addition, the disc/computer combination may include two disc-based software systems simultaneously (although only one can execute in core at a time): the Real-Time Executive System, and another software system (not the Time Shared Basic System). When loading into core from the disc with Basic Binary Disc Loader (BBDL), the operator specifies which system to load by setting switch register bit 0 equal to 0 (for RTE System) or 1 (for another disc-based system) after the BBDL halts. (See Section VII.)

When the two systems are generated, they must be stored on different areas of the disc. This is accomplished by protecting enough tracks to cover both systems; first generate the RTE System onto the initial tracks and then generate the other system on the remaining protected tracks. In this way, RTE does not attempt to write on the other system.

Another way to use the computer and disc for two or more software systems is to dump the RTE System on magnetic tape using SDUMP (see Section VII) before loading another system from magnetic tape.

APPENDIX C
SAMPLE RTGEN

```
DISC CHNL?
14

SYS DISC SIZE?
32

NO. PROTECTED?
Ø

# SECTORS/TRACK?
128

AUX DISC SIZE?
Ø

TBG CHNL?
12

PRIV. INT. CARD ADDR?
Ø

SWAPPING?
NO

LWA MEM?
37677

PRGM INPT?
PT

LIBR INPT?
PT

PRAM INPT?
TY

TURN OFF DISC PROTECT - PRESS RUN


*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
```

```
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT
*EOT


 NO UNDEF EXTS


PARAMETERS

/E


# OF BLANK ID SEGMENTS?
10

FWA BP LINKAGE?
30


SYSTEM

RTIOC(00)   02000
  EXEC      04671
  SCHED     06340

DVR00(00)   13444

DVR01(00)   14222

DVR02(00)   14547

DVR30(00)   14767


* EQUIPMENT TABLE ENTRY

13,DVR01
14,DVR30,D
16,DVR02,B
17,DVR00,B
11,DVR00
/E
```

```
* DEVICE REFERENCE TABLE

 1 = EQT #?
5
 2 = EQT #?
2
 3 = EQT #?
Ø
 4 = EQT #?
3
 5 = EQT #?
1
 6 = EQT #?
4
 7 = EQT #?
Ø
 8 = EQT #?
Ø
 9 = EQT #?
/E


* INTERRUPT TABLE

11,EQT,5
13,EQT,1
16,EQT,3
17,EQT,4
/E


LIBRARY

(NONE)

RT RESIDENTS

(NONE)

RT DISC RESIDENTS

(NONE)

FWA SY MEM 16771

CHANGE FWA SYS AV MEM?
25ØØØ

BG BOUNDARY?
26ØØØ
```

```
BG RESIDENTS

(NONE)

BG DISC RESIDENTS

LOADR(99)   26ØØØ

ASMB (99)   26ØØØ

ASMBD(99)   32425

ASMB1(99)   32425

ASMB2(99)   32425

ASMB3(99)   32425

ASMB4(99)   32425

ASMB5(99)   32425

FTN  (99)   26ØØØ

FTNØ1(ØØ)   26712
  SREAD     35Ø1Ø

FTNØ2(ØØ)   26712

FTNØ3(ØØ)   26712

FTNØ4(ØØ)   26712
  %WRIT     33256
  FADSB     3347Ø
  .FLUN     33633
  .PACK     33651

EDIT (99)   26ØØØ
  SREAD     3Ø371
  %WRIS     3Ø7Ø3


*SYSTEM STORED ON DISC
LAST SYS DISC ADDR: TRK Ø5 SEC Ø62(8)
```

APPENDIX D
SUMMARY OF OPERATOR REQUESTS

REQUEST	DESCRIPTION
ON,*name*[,NOW][*p1,p2,...p5*]	Schedules *name*. NOW means ignore time values.
OF,*name,p*	Terminates *name* (*p*=Ø). Purge *name* (*p*=8). Abort *name* (*p*>Ø).
SS,*name*	Suspend *name*.
GO,*name*[,*p1,p2,...p5*]	Re-schedule suspended *name*.
ST,*name*	Print status of *name*.
IT,*name*,*R*,*MPT*[,*HR*,*MI*,[,*SC*][,*MS*]]	Set time values of *name*.
PR,*name*,*n*	Set priority of *name*=*n*.
LG,*n*	Declare *n* L&G tracks or release L&G tracks (*n*=Ø).
LS,*p1*,*p2*	Declare disc number *p1*, track number *p2* of source file.
DN,*n*	Set EQT device *n* down.
UP,*n*	Set EQT device *n* up.
LU,*n*[,*m*]	Assign EQT *m* to LU *n*, release *n* (*m*=Ø), or print *n* (*m* absent).
EQ,*n*	Print EQT entry *n*.
EQ,*n*,*p*	Delete (*p*=Ø) or specify (*p*=1) buffering on EQT *n*.
TM,*day*,*hr*,*mi*,*sc*	Specify day of year, and 24 hour time.
TI	Print time.

APPENDIX E
SUMMARY OF EXEC CALLS

Consult Section IV for the complete details on each EXEC call. The general format of an EXEC call in assembly language is:

```
        EXT      EXEC        (Used to link program to RTE)
         .
         .
         .
        JSB      EXEC        (Transfer control to RTE)
        DEF      * + n + 1   (Defines point of return from RTE, n is
                             number of parameter's; must be direct address)
        DEF      P1  }       (Define addresses of parameters which may
         .           }       occur anywhere in program; may be multi-level
         .           }       indirect)
        DEF      Pn  }
        return point         (Continue execution of program)
         .
         .
         .
         .
P1      ---  }
 .           }
 .           }               (Actual parameter values)
 .           }
Pn      ---  }
```

For each EXEC call, this appendix includes only the parameters (P1 through Pn in the format above) of the assembly language calling sequence.

<u>READ/WRITE:</u> Transfers input or output.

RCODE	DEC	1(read) or 2(write)
CONWD	OCT	(See Section IV for control information.)
BUFFR	BSS	*n* (*n*-word buffer)
BUFFL	DEC	*n* or -2*n* (buffer length, words (+), characters (-))
DTRACK	DEC	*p* (optional disc track)
DSECT	DEC	*q* (optional disc sector)

I/O CONTROL:		Carry out control operations.
RCODE	DEC 3	
CONWD	OCT	(See Section IV for control information.)
PARAM	DEC	*n* (Optional parameter required by some CONWDs)

I/O STATUS:		Request device status.
RCODE	DEC *13*	
CONWD	DEC *n*	(Logical unit number)
STAT1	NOP	(Word 5 of EQT entry returned here)
STAT2	NOP	(Optional parameter for word 4 of EQT)

DISC ALLOCATION:		Request allocation of contiguous tracks.
RCODE	DEC 4	
#TRAK	DEC *n*	(Number of contiguous tracks desired) (If bit 15 = 1, do not suspend until available.)
STRAK	NOP	(Starting track returned here, or -1, not available.)
DISC	NOP	(Disc logical unit returned here.)
SECT#	NOP	(Sector returned here.)

DISC RELEASE:		Release some disc tracks assigned to the program.
RCODE	DEC 5	
#TRAK	DEC *n*	(If = -1, release all tracks, no more parameters; otherwise = number of tracks to release.)
STRAK	NOP	(Starting track number.)
DISC	NOP	(Logical unit.)

SUMMARY OF EXEC CALLS

<u>PROGRAM COMPLETION:</u> Signal end of program.

```
RCODE   DEC 6
```

<u>PROGRAM SUSPEND:</u> Suspend calling program.

```
RCODE   DEC 7
```

<u>PROGRAM SEGMENT LOAD:</u> Load segment of calling program.

```
RCODE   DEC 8
SNAME   ASC 3,xxxxx      (xxxxx is segment name)
```

<u>PROGRAM SCHEDULE:</u> To schedule another program.

```
RCODE   DEC              9 (waiting)or 1Ø(no waiting)
PNAME   ASC 3,xxxxx      (xxxxx is the program name)
P1      ...  ⎫
P2       .   ⎪
         .   ⎬           (Up to five optional parameters)
         .   ⎪
P5      ...  ⎭
```

<u>TIME REQUEST:</u> Request the 24-hour time and day.

```
RCODE   DEC 11
ARRAY   BSS 5            (Time values: tens of milliseconds, seconds,
                         minutes, hours, day, returned in that order)
```

EXECUTION TIME: Schedule a program by time.

Initial Offset Version

```
RCODE   DEC 12
PROG    NOP                (Schedule calling program) OR
        ASC 3,xxxxx        (Schedule xxxxx)
RESL    DEC x              (Resolution code)
MTPLE   DEC y              (Execution multiple)
OFSET   DEC-z              (x=units) equals the initial offset)
```

Absolute Start-Time Version

```
RCODE   (same)
PROG    (same as above)
RESL    (same)
MTPLE   (same)
HOURS   DEC a  ⎫
MINS    DEC b  ⎬           (Defines absolute start-time)
SECS    DEC c  ⎪
MSECS   DEC d  ⎭
```

APPENDIX F
ERROR MESSAGES

OPERATOR REQUEST ERROR MESSAGES

When an operator request is in error, RTE rejects the request and prints one of the messages below. The operator enters the request again, correctly.

Message	Meaning
OP CODE ERROR	Illegal operator request word.
NO SUCH PROG	The *name* given is not a main program in the system.
INPUT ERROR	A parameter is illegal.

EXEC CALL ERROR MESSAGES

When RTE discovers an error in an EXEC call, it terminates the program, releases any disc tracks assigned to the program, prints an error message on the operator console, and proceeds to execute the next program in the scheduled list.

When RTE aborts a program, it prints this message:

name ABORTED

When a memory project violation occurs that is not an EXEC call, this message is printed:

MP *name address (address* is the location that caused the violation.)

When an EXEC call contains an illegal request code, this message is printed:

RQ *name address* (*address* is the location that made the illegal call.)

The general error format, for other errors, is:

type name address

where	*type*	is a 4-character error code,
	name	is the program that made the call, and
	address	is the location of the call (equal to the exit point if the error is detected after the program suspends).

Error Codes for Schedule Calls

SCØ1 = Missing parameter,

SCØ2 = Illegal parameter,

SCØ3 = Program cannot be scheduled,
SCØ3 INT *xx* occurs when an external interrupt attempts to schedule a program that is already scheduled. RTE ignores the interrupt and returns to the point of interruption.

SCØ5 = Program given is not defined,

SCØ6 = No resolution code in EXECUTION TIME EXEC call.

Error Codes for Disc Allocation Calls

DRØ1 = Insufficient number of parameters,

DRØ2 = Number of tracks is zero, >255, or <zero; illegal logical unit; or number of tracks to release is zero or negative.

DRØ3 = Attempt to release track assigned to another program.

Error Codes for I/O Calls

IO∅1	=	Not enough parameters,
IO∅2	=	Illegal logical unit,
IO∅3	=	Logical unit not assigned,
IO∅4	=	Illegal user buffer,
IO∅5	=	Illegal disc track or sector,
IO∅6	=	Reference to a protected track; or using load-and-go before assigning load-and-go tracks (see LG, Section III),
IO∅7	=	Illegal read or write,
IO∅8	=	Disc transfer longer than track, and
IO∅9	=	Overflow of load-and-go area.

INPUT/OUTPUT ERROR MESSAGES

Illegal Interrupts

When an illegal interrupt occurs, RTE prints this message:

ILL INT *xx*

where *xx* is the octal channel number.

RTE clears the interrupt flag on the channel and returns to the point of interruption.

Equipment Messages

Message	Meaning
I/O ERR ET EQT #*n*	End-of-tape condition on device #*n*. Correct the condition and set device UP.

Continued on the next page.

Message	Meaning
I/O ERR NR EQT #*n*	Device #*n* is not ready. Make ready and set device UP.
I/O ERROR PE EQT #*n*	Parity error in data transmission from device #*n*. Examine device.
TRAK *nnn* EQT#*uu* S(or U)	Irrecoverable disc read parity error. If user read request (U), then program is abnormally terminated and track is made unavailable for further write operations. If system read request (S), the program transfer terminates.

FORTRAN COMPILER ERROR MESSAGES

This message is printed on the operation console, when an end-of-tape occurs on device #*n*.

```
I/O ERR ET EQT#
```

EQT #*n* is unavailable (see *DN*, Section III) until the operator declares it up:

```
UP,n
```

Thus, more than one source tape can be compiled into one program. The next source tape is loaded each time the compiler detects an End-of-Tape.

At the end of compilation, the following message is printed.

```
$END, FTN
```

and RTE executes the next scheduled program.

Two messages are I/O error messages generated by RTE when FTN attempts to write on the load-and-go tracks (RTE aborts FTN).

```
IOØ6
IOØ9
```

IOØ6 means that the load-and-go tracks were not defined (ie., by LG), and IOØ9 means that the load-and-go tracks overflowed. The operator must define more load-and-go tracks with LG and start compilation over again.

The compiler terminates abnormally if

- No source file is declared by LS, although logical unit 2 is given for input. Error E-ØØ19 is printed on the list device.
- The symbol table overflows. (E-ØØ14).

ALGOL MESSAGES

When the end of a source tape is encountered, the following message is output on the system teleprinter.

```
I/O ERR ET EQT #n
```

The compiler waits until the following message is input.

```
UP, n
```

If source input is indicated to be from the disc and the source pointer is not set, the diagnostic

```
NO SOURCE
```

is printed on the system teleprinter and compilation ceases.

At the end of a program, a program-termination request is made to the Executive. No message is printed. In case of a PAUSE statement, the following message is printed.

```
NAME:  PAUSE xxxx
```

where:	NAME = program name
	XXXX = a number which has no significance.

Execution is then suspended. To restart the program type:

```
GO,NAME[,P1,P2,P3,P4,P5]
```

$END ASMB PASS

The operator must replace the program in the input device and type

GO, ASMB

If an error is found in the Assembler control statement, the following message appears:

$END ASMB CS

If an end-of-file condition occurs before an END statement is found, the teleprinter signals

$END ASMB XEND

The current assembly stops.

If source input for logical unit 2 (disc) is requested, but no file has been declared (see *LS*, Section III), the teleprinter signals Assembly stops.

$END ASMB NPRG

RTE generates two messages when ASMB attemps to write on the load-and-go tracks (ASMB aborted).

IO∅6
IO∅9

IO∅6 means than the load-and-go tracks were never defined by an LG operator request, and IO∅9 means that the load-and-go tracks have overflowed.

The next message is associated with each error diagnostic printed during pass 1.

nnn

nnn is the "tape" number where the error (reported on the next line of the listing)occurred. A program may consist of more than one tape. The tape counter starts with one and increments whenever an End-of-Tape (paper tape) or a blank card is encountered. When the counter increments, the numbering of source statements starts over at one.

Each error diagnostic printed during pass 2 of the assembly is associated with a different message:

PG *ppp*

ppp is the page number (in the listing) of the *previous* error diagnostic. PG ØØØ is associated with the first error in the program.

The messages (#*nnn* and PG *ppp*) occur on a separate line, just above each error diagnostic in the listing.

EDITOR ERROR MESSAGES

RTE Editor prints error messages on the operator console in this format:

/EDIT: *error message: illegal edit command*

Then the RTE Editor continues with the Edit File; there is no on-line correction of illegal edit commands.

Error Message	Meaning
MEM OVERFLOW	The Edit File overflows available memory; RTE Editor prints the command causing the overflow. Edit terminates.
CS ERR	Illegal edit command, which is printed.

Continued on the next page.

Error Message	Meaning
PARAM ERR	Edit command "r" or "c" is illegal: non-numeric, = Ø, >72, $r_2 \leq r_1$, $c_2 \leq c_1$; command printed.
SEQ ERR	"r" parameter $\leq$ a previous "r", or "r" greater than range of Symbolic File; command printed.
/I ERR	No insert source statements after /I; command printed.
/R ERR	No replacement statements after /R; command printed.
/C OVF	Character overflow in edit statement (ie. >72 character)
DISK OVF	No disc space for file; edit terminates.
FILE UN	Undefined Symbolic File, or LUN (Edit File) = 2. Edit terminates.

RELOCATING LOADER ERROR MESSAGES

Error messages are printed in this format:

/LOADR: *message*

LØ1 and LØ2 (checksum error and illegal record) are recoverable errors. The offending record can be re-read by repositioning the tape and typing:

GO,LOADR

For irrecoverable errors, one of the following messages is printed, followed by "LOADING ABORTED" (the loader is terminated)"

Continued on the next page.

L∅3 - Memory overflow,

L∅4 - Base page linkage area overflow,

L∅5 - Symbol table area overflow,

L∅6 - Common block error,

- Exceeding allocation in a replacement or addition,
- In a normal background load, first program did not declare largest Common block,

L∅7 - Duplicate entry points,

L∅9 - Record out of sequence.

L∅8 means there was no transfer address (main program) in the program unit. Another program may be entered with a GO operator request. (This also occurs when load-and-go is specified, but no program exists in the load-and-go area).

L1∅ also means that an error was found in an operator request parameter. GO requests may be retyped; ON requests may not.

NO BLANK ID SEGMENTS is printed when an available (i.e., blank) ID Segment is not found. The loader calls for program suspension. The operator may then delete a program from the system (OF operator request) or may terminate the loader.

WAITING FOR DISC SPACE is printed when a track allocation cannot be made. The loader repeats the disc request and is suspended until space becomes available. This message is primarily for information, as no action is required.

UNDEFINED EXTS is printed followed by a list of all remaining undefined external symbols after a scan of the library. Additional programs may be loaded by the GO operator request.

LOAD is printed and the loader is suspended whenever an End-Of-Tape condition is detected from the input unit.

DUPLICATE PROG NAME - *name* is printed when a program *name* is already defined in the system for a normal load or a program addition. The loader changes the name of the current program by replacing the first two characters with "##".

If the parameter word equals zero, the automatic single space is to be suppressed on the next print operation only.

CARRIAGE CONTROL CHANNELS

If the parameter word is between 55 and 64, then the printer spaces using the carriage control channels, which have the following meanings:

Channel 1	Single space with automatic page eject,
Channel 2	Skip to next even line with automatic page eject,
Channel 3	Skip to next triple line with automatic page eject,
Channel 4	Skip to next 1/2 page boundary,
Channel 5	Skip to next 1/4 page boundary,
Channel 6	Skip to next 1/6 page boundary,
Channel 7	Skip to bottom of the page,
Channel 8	Skip to top of next page.

AUTOMATIC PAGE EJECT

During non-automatic page eject mode, if the parameter word is equal to 56, then it is interpreted as equal to 1. Automatic page eject mode applies only to single space operations.

02116-9139