HEWLETT PACKARD

RTE-C Software System

Programming and Operating Manual

RTE-C
Software System
Programming and Operating Manual

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

PART NO. 29101-93001

Printed in U.S.A. 01/76

LIST OF EFFECTIVE PAGES

Changed pages are identified by a change number adjacent to the page number. Changed information is indicated by a vertical line in the outer margin of the page. Original pages do not include a change number and are indicated as change number 0 on this page. Insert latest changed pages and destroy superseded pages.

Change 0 (Original) ..... Jan 1976

CONTENTS

ILLUSTRATIONS

TABLES

SECTION I
GENERAL DESCRIPTION

INTRODUCTION

The HP Real-Time Core-Based Software System (RTE-C) is a multiprogramming system that allows several programs to operate concurrently, each program executing during the unused central processor time of the others. Other features the RTE-C System provides are:

- Generalized priority scheduling of real-time and general-purpose operations.
- Centralized control of all input/output operations and associated interrupt processing.
- Modular organization of the major programs at the source level to facilitate the generation of a system tailored to a user's requirements.
- On - line loading or replacement of absolute programs which have been prepared by the RTE-C Relocating Loader.

SOFTWARE

Table 1-1 is a list of the software that makes up the RTE-C System. Also included in the table are other software modules appropriate to system operation. System Input/Output (SIO) Drivers are not included.

Table 1-1. RTE-C Software

Description	Binary Tape Number
Executive (EXEC)	29101-60001
Scheduler (SCHED)	29101-60002
I/O Control (RTIOC)	29101-60003
Absolute Program Loader (APLDR)	29101-60004
Relocating Loader	29101-60010
System Generator (RTSGN)	29101-60011
RTE/DOS FORTRAN Formatter	24153-60001
RTE/DOS Relocatable Library, EAU	24151-60001
RTE/DOS FORTRAN IV Library	24152-60001
RTE/DOS Floating Point Library	24248-60001
HP Assembler, EAU	25117-60574
Symbolic Editor	20100-60001
HP FORTRAN Compiler Pass 1	25117-60289
HP FORTRAN Compiler Pass 2	25121-60014
Multiple Device Driver DVROO	29029-60001

HARDWARE

RTE-C operates in the following minimum hardware configuration.

HP 2100S Computer with 8K memory.

Time Base Generator.

Keyboard device (e.g. teleprinter) for operator/system. Communication and control.

Punched Tape Reader.

Tape Punch (optional).

SYSTEM DESCRIPTION

RTE-C is a multiprogramming system. All input/output and interrupt processing is controlled by RTE-C, except for special privileged interrupts which circumvent RTE-C for quicker response. When a program requests a non-buffered I/O transfer, RTE-C 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-C reschedules the suspended program for execution. User programs can be written in Assembly, or FORTRAN Languages. Programs are scheduled by time intervals, an external device, an operator request, or by another program. RTE-C has a scheduling module which decides when to execute the competing programs.

SUBROUTINES

Each user program (main or segment) consists of a primary routine, containing the transfer point for entry into the program from RTE-C, and optionally, 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. The primary routine is linked with its subroutines (which are defined by external references within the primary routine) when it is loaded.

The Relocatable Library consists of a number of subroutines that may be linked to user programs. (See Appendix D). Each subroutine is either re-entrant, privileged, or utility. These terms are defined as follows:

Re-entrant – Can be interrupted
Privileged – Cannot be interrupted
Utility – Used by one program only

RE-ENTRANT/PRIVILEGED SUBROUTINE–A re-entrant or privileged subroutine may be used by more than one program at a time. The user gathers all the routines of this class together and loads them into the resident library area during Real-Time System Generation (RTSGN). If a re-entrant or privileged routine is to be used by only one program, the routine could be appended to the calling program as a utility routine. In this case, since the routine is not in the resident library area, it cannot be shared by any other programs.

UTILITY SUBROUTINE–Subroutines which cannot be shared because of internal design or I/O considerations are utility subroutines. A copy of the utility subroutine is appended to each program that calls it when the program is relocated during the generation process, or relocated with the RTE-C Relocating Loader.

PROGRAM SCHEDULING (SCHED)

Scheduling of all programs is done by the scheduling module SCHED, and is based on priority. Programs may be scheduled for execution by an operator request, a program request, a device interrupt, or the completion of a time interval. The RTE-C System can be generated such that one program is automatically scheduled each time the system is loaded into core. Whenever programs conflict because of simultaneous demands for execution, SCHED decides in favor of the highest priority program. Priorities are assigned by the user during RTSGN or on-line loading, and may be changed by an operator request.

The RTE-C System handles priorities on a completely generalized basis. The highest priority program scheduled for 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 scheduled list until a higher priority program is rescheduled.

Programs that were removed from the executing state to wait for an event to occur before re-scheduling are in the suspended state. Programs which are not currently in either the scheduled, executing, or suspended state are in the dormant state. Programs may thus be in one of four states:

Executing

Scheduled

Suspended

Dormant

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

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 program scheduled by the suspended program

The operator has requested that a program be suspended

The program has requested that it be suspended.

Program priority determines the order of a program in the scheduled and suspended 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 can be changed by an operator request, and more than one program can be at the same priority.

For each program state, except executing, RTE-C 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 B) 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 three types of suspension lists:

I/O suspension lists (one for each device)

Memory availability list

Operator suspension list

INPUT/OUTPUT CONTROL (RTIOC)

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

INTERRUPT PROCESSING—All interrupts, except privileged interrupts, cause a transfer to the central interrupt control (CIC) in RTIOC, 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, contains a pointer to the correct processor routine for each interrupt. 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.

INPUT/OUTPUT PROCESSING—RTIOC 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 for a particular device 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 RTIOC 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. RTIOC handles the request. It suspends the requesting program until the operation is complete unless the request is for output to a buffered device. All input/output operations occur concurrently with program execution, however, if the transfer is non-buffered, the requesting program is suspended and the next lower priority scheduled program is allocated execution time during the suspension.

EXEC COMMUNICATION (EXEC)

When an executing program makes an EXEC call, it attempts to execute a JSB to EXEC in the protected area of core. This causes a memory protect violation interrupt, which CIC recognizes, and transfer to EXEC. EXEC examines the parameters associated with the JSB in the calling program. If the parameters are legal, EXEC either handles the request itself or goes to the appropriate part of RTE-C to process it.

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

Perform input and output operations.

Terminate or suspend itself.

Schedule other programs.

Obtain the time of day.

Set execution time cycles.

OPERATOR REQUESTS

The operator retains ultimate control of the RTE-C System with requests entered through the teleprinter keyboard. (See Section II.) Operator requests, which are handled by SCHED, can interrupt RTE-C 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.

Load absolute programs on-line.

Declare I/O devices up or down.

Dynamically alter the logical I/O structure and buffering designations.

Eliminate or replace programs in the system.

Examine and dynamically alter an I/O device's time-out parameter.

List information from ID segments about programs.

Initialize the real-time clock and print the time.

SYSTEM CONFIGURATION

User programs, system programs, library routines, and Real-Time Executive Modules are incorporated into a

configured RTE-C System. The RTE-C software is modular and of a general nature, so the user can configure his particular programs and I/O device drivers into a real-time system tailored to his exact needs.

Using the Real-Time System Generator (RTSGN), the relocatable software modules and user programs are converted into a configured real-time system in absolute binary format which is punched on paper tape. In operation, the configured system is loaded into the computer with the Basic Binary Loader.

RTE-C SYSTEM SUMMARY

The Hewlett-Packard Real-Time Executive Core Based Software System is a multiprogramming, system with priority scheduling, interrupt handling, and on-line program loading capabilities.

With multiprogramming, a number of data acquisition systems or test stands can be operated simultaneously on a 24-hour a day basis. Data reduction and report preparation functions can be scheduled to execute during times when real-time activities permits.

When the RTE-C System is not being used to run real-time programs, the computer can be used by the programming group for development work with compilers for FORTRAN, the HP Assembler, Editor, and the relocation of user programs for future loading.

Scheduling of all programs is based on priority. External events can interrupt to schedule programs for execution, or a program can be scheduled by an operator request, a program request, or on a real-time clock basis. Priorities may be assigned by the user during RTSGN if they weren't included during program development. Priorities may also be changed by an operator request.

The Executive controls I/O processing through a central routine that directs requests and interrupts to the appropriate device driver subroutine. For efficiency, programs awaiting I/O are suspended to let other programs use the computer. Outputs to slow devices can be buffered. For processes that cannot tolerate ordinary system overhead, a privileged interrupt option lets a device contact its driver directly without going through the Executive.

The operator retains ultimate control of the RTE-C System with requests entered through the teleprinter keyboard. The operator can turn programs on, make status checks, or perform other operations.

Configuration is efficient. The generation program RTSGN, in a dialog with the operator, configures the software for a particular hardware system on that system itself.

SECTION II
OPERATOR REQUESTS

INTRODUCTION

The operator controls an executing Real-Time Executive System by operator requests entered through the teleprinter console. These operator requests can interrupt RTE-C to perform the functions described in Table 2-1. Included in the table and this section are three commands associated with the absolute program loader (APLDR). These commands (LO, PL, and RP) are available only if the APLDR was included in the system generation. When any one of the three commands are entered, RTE-C automatically schedules and runs APLDR.

COMMAND STRUCTURE

The operator gains the attention of RTE-C by pressing any key on the console. When RTE-C responds with an asterisk (*), the operator types any operator request, consisting of a two-character request word (e.g., ON, UP, etc.) and the appropriate parameters separated by commas. Each command is parsed, or resolved, by a central routine that accepts certain conventions. Command syntax is described in Table 2-2 and, with the conventions described next, must be followed exactly to satisfy system requirements.

COMMAND CONVENTIONS

- When the data is entered, the items outside the brackets are required symbols, and the items inside the brackets are optional. Note that when RTE-C is restarted, any parameters previously changed are restored to their original value set during RTSGN.

- If an error is made in entering the parameters, CONTROL and A struck simultaneously will delete the last character entered if input is a teletype. If the input is through the HP 2600 CRT terminal, the last character can be deleted with BACKSPACE. To delete the entire line use RUBOUT followed by carriage return/line feed. Each request must be completed with an end-of-record terminator (e.g., carriage return/line feed for the teleprinter and CRT).

- Two commas in a row mean a parameter is zero.

Table 2-1. RTE-C Operator Commands

Command Format	Description
DN	Declare I/O device unavailable.
EQ	Examine the status of any I/O device, and dynamically alter device buffering assignments.
GO	Restart programs out of suspension.
IT	Schedule programs to execute at specified times.
LO	Load absolute program on-line using APLDR.
LU	Dynamically alter device logical unit assignments.
OF	Turn programs off.
ON	Turn programs on.
PL	List information from ID segments about programs.
PR	Change the priority of dormant programs.
RP	Replace named absolute program with a new one.
SS	Suspend programs.
ST	Examine the status of programs.
TI	Print the current time.
TM	Set the real-time clock.
TO	Examine and dynamically alter an I/O device's time-out parameter.
UP	Declare I/O devices available.

Table 2-2. Conventions in Operator Command Syntax

Item	Meaning
UPPER CASE ITALICS	These words are literals and must be specified as shown.
lower case italics	These are symbolic representations indicating what type of information is to be supplied. When used in text, the italics distinguishes them from other textual words.
[,*item*]	Items with brackets are optional. However, if *item* is not supplied, its position must be accounted for with a comma; this causes *item* to automatically default.
. . . . (row of dots)	This notation means "and so on."

DN

Purpose:

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

Format:

DN, *n*

Where:

n	is in the EQT entry number of the I/O device to be set down.

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.

EQ, *n*

Purpose:

To print the description and status of an I/O device, as recorded in the EQT entry.

Format:

EQ, *n*

Where:

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

COMMENTS

The information is printed as:

select code DVR*nn* D B U*nn* status

Where:

select code	is the I/O channel.
DVR*nn*	is the driver routine
D	is D if DMA required, 0 if not,
B	is B if automatic output buffering used, 0 if not,
U*nn*	is the last subchannel addressed
status	is the logical status: 0 - 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.

Format:

EQ, *n, p*

Where:

n	is the EQT entry number of the I/O device.
p	is 0 to delete buffering, or
p	is 1 to specify buffering.

COMMENTS

When the system is reloaded into core, buffering designations made by EQ, *n, p* are reset to the values originally made by RTSGN.

GO

Purpose:

To reschedule a program that has been suspended by an SS operator request or a Suspend EXEC Call.

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*. Note, these parameters should be used only when the program has suspended itself through the Suspend EXEC Call.

COMMENTS

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

NOTE

Do not enter parameters if the program was suspended by an SS operator request lest the saved contents of the B-Register be destroyed.

When a program resumes execution after suspending itself, the address of the parameters passed by GO is in the B-Register. In FORTRAN, an immediate call to the library subroutine RMPAR retrieves the parameters. (See Section III, Suspend EXEC Call.) If no parameters are entered (program suspended by SS operator request), the B-Register is restored to its value before suspension.

IT

Purpose:

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

Format:

IT, *name*,r, *mpt*[, *h* ,*min* [, *s*[, *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 0 to 999 which is used with *r* to give the actual time interval for scheduling (see Comments).	
h	hours	
min	minutes	sets an initial start time.
s	seconds	
ms	tens of ms.	

COMMENTS

The resolution code (*r*) is the units in time to be multiplied by the multiple execution interval value (*mpt*) to get the total time interval. Thus, if *r* = 2 and *mpt* = 100, *name* would be scheduled every 100 seconds. If *h, min, s,* and *ms* are present, the first execution occurs at the initial start time which these parameters specify. (Program must be initialized with ON operator request.) If *h, min, s,* and *ms* are not present, the program can be called by another program, or started with the ON, *NOW* operator request.

When the system is reloaded into core, time values set by IT are lost, and the original time values set during RTSGN are reinstated.

The IT operator request is similar to the Execution Time EXEC Call. (See Section III.)

LO

Purpose:

To load an absolute program into the operating system and build an ID segment for it.

Format:

LO [, *name* [, *lu*]]

Where:

name	is the name to be given to the program. *Name,* if given, overrides the name given to the program during preparation.
lu	is the logical unit number for the absolute binary input. Logical unit 5 is default.

COMMENTS

The program must be in absolute binary code to be accepted by the APLDR for on-line loading. Refer to Section VI for information on using the relocating loader to prepare an absolute program for on-line loading. If the program being loaded is going to interfere with an existing program (or place it in jeopardy; i.e., overlay and destroy all or a part of the program), an error occurs which is reported as:

APLDR: REM *name*

where

name	is the name of the program that is being placed in jeopardy.

The operator can remove the jeopardized program and continue the loading process by issuing the commands:

OF, *name, 8*
GO, *APLDR*

where

name	is the name of the program placed in jeopardy, and to be permanently removed from the system.

The GO, *APLDR* operator request re-schedules the ADLDR to continue loading the new program.

LU

Purpose:

To print or change a logical unit number assignment.

Format:

LU, *n* [, *m* [, *p*]]

Where:

n	is a logical unit number from 1 to 63_{10}.
m	if present, is an EQT entry number to assign to *n*.
m	if zero (0), releases logical unit number.
p	if present, is a subchannel number (0-7) to assign to *n*

If *m* and *p* are absent the assignment of logical unit *n* is printed.

COMMENTS

The information is printed as

LU # 07 = # 05, U02

which means logical unit number 7 is assigned to EQT entry number 5, subchannel number 2.

LU1 must be a keyboard entry device (e.g., teleprinter or CRT). If LU1 is changed from one keyboard device to another, the new device will print a double asterisk (**). LU2 (system disc) and LU3 (aux disc) cannot be changed to a new EQT entry number (i.e., if *n* = 2 or 3, no other parameter can be entered). These are reserved for compatibility with RTE disc based systems.

When an irrecoverable problem occurs on an I/O device, the operator can bypass the downed device for future requests 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.

When the system is reloaded into core, any assignments made by LU are reset to those originally set by RTSGN.

Section V, Real-Time Input/Output, explains logical unit numbers, equipment table entry numbers, and subchannel numbers in detail.

OF

Purpose:

To terminate a program or to completely remove a program from the RTE-C system.

Format:

OF, *name, p*

Where:

name	is the name of a program,
$p = 0$	terminates and removes from the time list any executing, scheduled, or operator suspended program; terminates programs which are I/O, or memory suspended the next time they are scheduled.
$p > 0, \neq 8$	terminates immediately the program named, and removes it from the time list. If suspended for I/O, the device and channel are cleared by a CLC.
$p = 8$	same as for $p = 0$, plus, if the program is not I/O suspended, the program is completely removed from the RTE-C system. If the program is I/O suspended, the OF request is treated as if p were greater than 0, but not equal to 8. OF, *name*, 8 must be entered again to permanently remove *name* from the system. The ID segment is blanked, and then made available for loading another program.

ON

Purpose:

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

Format:

ON, *name* [, *NOW*] [, *p1* [, ... [*p5*]]]]]

Where:

name	is the name of a program,
NOW	schedules a program immediately that is normally scheduled by the clock time (see IT).
p1 through *p5*	are parameters passed to the program when it is scheduled (must be positive integers less than 32767).

COMMENTS

The parameters *p1* through *p5* are the ones passed by RMPAR as described in Comments under the Program Schedule EXEC call in Section III.

If the resolution code in the ID segment of the program is not zero, RTE-C 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). The resolution code may be non-zero as a result of:

a. Generation
   1. With a resolution code in the *name* record
   2. Entry of a resolution code during parameter input phase.

b. The IT command.

c. Scheduling the program with the clock by some program in the system.

PL

Purpose:

To list all information about program from the ID segments.

Format:

PL[, *lu*]

Where:

lu	is the logical unit number of the list device. Logical unit 6 is default.

COMMENTS

The listing begins with the heading

PROGRAM LIST: NAME, PRIORITY MAIN, BASE PAGE

name priority xxxxx xxxxx yyyyy yyyyy

.

.

.

nn BLANK ID SEGMENTS

Where:

name	is the name of a program
priority	is the program's priority.
xxxxx	is the low and high memory bounds of the program.
yyyyy	is the low and high base page memory bounds of the program.

ID segments which are not currently being used by a program are counted and reported at the end of the listing.

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

One is the highest priority, and 99 is the lowest. Only dormant programs can have their priority changed. When the system is reloaded into core, the priority of *name* resets to the value set by RTSGN.

RP

Purpose:

To replace an absolute program that is already in core. The new program will use the old program's ID segment.

Format:

RP, *name* [, *lu*]

Where:

name	is the name of the program to be replaced.
lu	is the logical unit number for the absolute binary input. Logical unit 5 is default.

COMMENTS

The new program uses the old program's ID segment; therefore the old program is no longer accessible after a replacement. This also means that the new program may or may not occupy the same core locations as the old program. This is because the old program's ID segment no longer points to the old program. The RP operator request is the same as:

OF, *name, 8*
LO

The old program may not be replaced if it is not dormant, has a non-zero point of suspension, or is in the time list. If any of these conditions are true the following error message results:

OF, *name*

This message indicates that the program cannot be replaced because it is not dormant, etc. Use the following commands to abort the old program and continue the replacement.

OF, *name, 1*
GO, *APLDR*

To abort the entire process, enter OF, *APLDR, 1.*

If the program being loaded is going to interfere with an existing program (or place it in jeopardy), an error occurs which is reported as:

APLDR: REM *name*

where:

name	is the name of the program that is being placed in jeopardy.

The operator can remove the jeopardized program and continue the loading process by issuing the commands:

OF, *name,8*
GO, *APLDR*

where:

name	is the name of the program placed in jeopardy, and to be permanently removed from the system.

The GO, *APLDR* operator request re-schedules the APLDR to continue loading the new program.

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 for I/O, or memory, RTE-C waits until the current suspend is over, then suspends the program with SS.

The SS operator request is similar to the Program Suspend EXEC Call. (See Section III.)

ST

Purpose:

To request the status (priority, current list, time values) of a program

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 fix format:

PR S R MPT H MIN S MS T

Where:

PR is the priority, a value from 1 to 99_{10},
S is the current list in which the program is located:
0 - Dormant
1 - Scheduled
2 - I/O suspend
3 - Not used
4 - Unavailable memory suspend
5 - Not used
6 - Operator suspend or programmed suspend (EXEC 7 Call)

R, MPT, H, MIN, S and MS are all zero (0) 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).

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 H MIN S

Where:

DAY is the three-digit day of the year and
H, MIN, S is the time on a 24-hour clock. (See Table 2-5 for day of year conversion.)

The TI operator request is similar to the Time Request EXEC Call (See Section III.)

TM

Purpose:

To set the real-time clock

Format:

TM, *day, h, min, s*

Where:

day is a three-digit day of the year (see Table 2-5),
h, min, and *s* is the current-time on a 24 hour clock.

COMMENTS

The operator usually gives TM in response to the message printed when the RTE-C system is initiated:

SET TIME

The response sets the time when the line-feed key is pressed. Enter a time value ahead of real-time and press carriage-return. When real-time equals the entered value, press line-feed. The system is now synchronized with the time of day.

NOTE

The RTE-C clock is automatically started from time=zero each time the system is loaded into core.

TO

Purpose:

To print or change the time-out parameter of an I/O device.

Format:

TO, *n*[, *m*]

Where:

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

m is the number of 10 ms intervals to be used as the time-out value. (*m* cannot be less than 500 (5 sec) for the system input device.)

If *m* is absent the time-out value of EQT *n* is printed.

COMMENTS

The information is printed as

TO #03 = 0100

which means EQT entry number 3 has a time-out value of one second. The time-out value is calculated using *m* time base generator interrupts (the time base generator interrupts once every 10 ms). For example, *m* = 100 sets a time-out value of one second: 100 times 10 ms equals 1 second.

DRIVER PROCESSING OF TIME-OUT

A driver indicates to the system that it wants to process time-out by setting bit 12 in EQT word 4. The system never clears this bit so it need be set only once. In this case, when a device times out, the following events take place:

- Bit 11 in EQT word 4 is set.
- The driver is entered at C.XX with the A-register set to the select code (from EQT word 4).
- The driver must recognize that the entry is for time-out by examining bit 11 of EQT word 4 and do whatever is necessary. The driver should then clear bit 11 in the event it is entered again prior to completion of the operation so that it knows why it is being entered on the next call. (RTIOC) will clear this bit prior to entering the driver at I.XX.)
- The driver may continue or complete the operation. If it completes the operation it may set the A-Register to 4 to indicate time-out.
- If the A-Register is set to 4, RTIOC will set the indicated device down and issue the following message.

I/O ERROR TO EQT *n*

SYSTEM PROCESSING OF TIME-OUT

In the case where the driver does not set bit 12 of EQT word 4, the following events take place on time-out:

- The calling program is rescheduled, and a zero transmission log is returned to it.
- The device is set to the down status, and bit 11 in the fourth word of the device's EQT entry is set to one. The following error message is printed:

I/O ERROR TO EQT *n*

- The system issues a CLC to the device's I/O select code(s) through the EQT number located in the interrupt table.

The device set down is unavailable until set up by the UP operator request. When the system is reloaded into core, time-out values set by TO are reset to the values originally set during RTSGN.

UP

Purpose:

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

Format:

UP, *n*

Where:

n	is the EQT entry number of the device to be set up.

COMMENTS

When the operator or the RTE-C 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 for future requests (see LU, this section). However, previous requests made to this device are not switched to the new device. To prevent indefinite I/O suspension on a downed device time-out is used. Refer to I/O Device Time-Out in Section V.

ERROR MESSAGES

Error messages are generated by RTE-C in association with entering operator requests, and on-line program loading with APLDR. When one of the three operator requests associated with APLDR (LO, PL, and RP) is entered, RTE-C automatically schedules and runs APLDR. For this reason the error messages are separated into those from standard operator requests, and those from the APLDR.

OPERATOR REQUEST ERROR MESSAGES

When a request is in error, RTE-C prints one of the messages shown in Table 2-3.

APLDR ERROR MESSAGES

When the APLDR is running, it alerts the operator to errors and prints the messages shown in Table 2-4.

Table 2-3. Operator Request Error Messages

Message	Meaning
OP CODE ERROR	Illegal operator request word.
NO SUCH PROG	The name given is not a program in the system.
INPUT ERROR	A parameter is illegal.
ILLEGAL STATUS	Program is not in appropriate state.

Table 2-4. APLDR Generated Error Messages

Message	Meaning
APLDR:	Precedes all error messages from APLDR.
00 BLANK ID SEGMENTS	No ID Segments available to add a program –Use the RP Command to reuse an ID Segment.
NO *name*	There is no ID Segment for the associated program name. –Check spelling of the name.
ABORTED	The Absolute Program Loader aborts if any of the above error messages are given.
REM *name*	*name* is a program which is occupying some core needed to load the new program. –Use the OF, *name, 8* command to remove the program from the system and then continue loading by GO, *APLDR*. To abort the process type OF, *APLDR, 1*.
OF, *name*	*name* is the name of a program specified in the RP Command. This message indicates that the program cannot be replaced because it is not dormant, or it has a non-zero point of suspension. –Use the OF, *name, 1* command to abort the program, then enter GO, *APLDR* to continue the replacement (RP Command).
DUP *name*	*name* is a previously loaded program. APLDR will try to replace the first two characters of the name with "$$". If it is still not unique, APLDR will abort.
The following are errors detected by APLDR related to the absolute code being entered. APLDR is aborted and no new ID segment is created.	
CKSM	Checksum error was detected. –Must revise absolute program.
COM	The absolute program uses common storage which is not within the bounds of the real-time program area.
MEM	The absolute program contains code which does not fit within the bounds of the available real-time program area.
ID?	The special absolute records required by the APLDR to build an ID Segment could not be found. The special records are automatically produced by the RTE-C Relocating Loader.

Table 2-5. Day of Year

JANUARY						
	1/2 (2)	1/3 (3)	1/4 (4)	1/5 (5)	1/6 (6)	1/7 (7)
1/8 (8)	1/9 (9)	1/10 (10)	1/11 (11)	1/12 (12)	1/13 (13)	1/14 (14)
1/15 (15)	1/16 (16)	1/17 (17)	1/18 (18)	1/19 (19)	1/20 (20)	1/21 (21)
1/22 (22)	1/23 (23)	1/24 (24)	1/25 (25)	1/26 (26)	1/27 (27)	1/28 (28)
1/29 (29)	1/30 (30)	1/31 (31)				

FEBRUARY						
2/1 (32)	2/2 (33)	2/3 (34)	2/4 (35)	2/5 (36)	2/6 (37)	2/7 (38)
2/8 (39)	2/9 (40)	2/10 (41)	2/11 (42)	2/12 (43)	2/13 (44)	2/14 (45)
2/15 (46)	2/16 (47)	2/17 (48)	2/18 (49)	2/19 (50)	2/20 (51)	2/21 (52)
2/22 (53)	2/23 (54)	2/24 (55)	2/25 (56)	2/26 (57)	2/27 (58)	2/28 (59)
2/29 (60)	LEAP YEAR ONLY					

MARCH						
3/1 (60)	3/2 (61)	3/3 (62)	3/4 (63)	3/5 (64)	3/6 (65)	3/7 (66)
3/8 (67)	3/9 (68)	3/10 (69)	3/11 (70)	3/12 (71)	3/13 (72)	3/14 (73)
3/15 (74)	3/16 (75)	3/17 (76)	3/18 (77)	3/19 (78)	3/20 (79)	3/21 (80)
3/22 (81)	3/23 (82)	3/24 (83)	3/25 (84)	3/26 (85)	3/27 (86)	3/28 (87)
3/29 (88)	3/30 (89)	3/31 (90)				

APRIL						
4/1 (91)	4/2 (92)	4/3 (93)	4/4 (94)	4/5 (95)	4/6 (96)	4/7 (97)
4/8 (98)	4/9 (99)	4/10 (100)	4/11 (101)	4/12 (102)	4/13 (103)	4/14 (104)
4/15 (105)	4/16 (106)	4/17 (107)	4/18 (108)	4/19 (109)	4/20 (110)	4/21 (111)
4/22 (112)	4/23 (113)	4/24 (114)	4/25 (115)	4/26 (116)	4/27 (117)	4/28 (118)
4/29 (119)	4/30 (120)					

MAY						
5/1 (121)	5/2 (122)	5/3 (123)	5/4 (124)	5/5 (125)	5/6 (126)	5/7 (127)
5/8 (128)	5/9 (129)	5/10 (130)	5/11 (131)	5/12 (132)	5/13 (133)	5/14 (134)
5/15 (135)	5/16 (136)	5/17 (137)	5/18 (138)	5/19 (139)	5/20 (140)	5/21 (141)
5/22 (142)	5/23 (143)	5/24 (144)	5/25 (145)	5/26 (146)	5/27 (147)	5/28 (148)
5/29 (149)	5/30 (150)	5/31 (151)				

JUNE						
6/1 (152)	6/2 (153)	6/3 (154)	6/4 (155)	6/5 (156)	6/6 (157)	6/7 (158)
6/8 (159)	6/9 (160)	6/10 (161)	6/11 (162)	6/12 (163)	6/13 (164)	6/14 (165)
6/15 (166)	6/16 (167)	6/17 (168)	6/18 (169)	6/19 (170)	6/20 (171)	6/21 (172)
6/22 (173)	6/23 (174)	6/24 (175)	6/25 (176)	6/26 (177)	6/27 (178)	6/28 (179)
6/29 (180)	6/30 (181)					

JULY						
7/1 (182)	7/2 (183)	7/3 (184)	7/4 (185)	7/5 (186)	7/6 (187)	7/7 (188)
7/8 (189)	7/9 (190)	7/10 (191)	7/11 (192)	7/12 (193)	7/13 (194)	7/14 (195)
7/15 (196)	7/16 (197)	7/17 (198)	7/18 (199)	7/19 (200)	7/20 (201)	7/21 (202)
7/22 (203)	7/23 (204)	7/24 (205)	7/25 (206)	7/26 (207)	7/27 (208)	7/28 (209)
7/29 (210)	7/30 (211)	7/31 (212)				

AUGUST						
8/1 (213)	8/2 (214)	8/3 (215)	8/4 (216)	8/5 (217)	8/6 (218)	8/7 (219)
8/8 (220)	8/9 (221)	8/10 (222)	8/11 (223)	8/12 (224)	8/13 (225)	8/14 (226)
8/15 (227)	8/16 (228)	8/17 (229)	8/18 (230)	8/19 (231)	8/20 (232)	8/21 (233)
8/22 (234)	8/23 (235)	8/24 (236)	8/25 (237)	8/26 (238)	8/27 (239)	8/28 (240)
8/29 (241)	8/30 (242)	8/31 (243)				

SEPTEMBER						
9/1 (244)	9/2 (245)	9/3 (246)	9/4 (247)	9/5 (248)	9/6 (249)	9/7 (250)
9/8 (251)	9/9 (252)	9/10 (253)	9/11 (254)	9/12 (255)	9/13 (256)	9/14 (257)
9/15 (258)	9/16 (259)	9/17 (260)	9/18 (261)	9/19 (262)	9/20 (263)	9/21 (264)
9/22 (265)	9/23 (266)	9/24 (267)	9/25 (268)	9/26 (269)	9/27 (270)	9/28 (271)
9/29 (272)	9/30 (273)					

OCTOBER						
10/1 (274)	10/2 (275)	10/3 (276)	10/4 (277)	10/5 (278)	10/6 (279)	10/7 (280)
10/8 (281)	10/9 (282)	10/10 (283)	10/11 (284)	10/12 (285)	10/13 (286)	10/14 (287)
10/15 (288)	10/16 (289)	10/17 (290)	10/18 (291)	10/19 (292)	10/20 (293)	10/21 (294)
10/22 (295)	10/23 (296)	10/24 (297)	10/25 (298)	10/26 (299)	10/27 (300)	10/28 (301)
10/29 (302)	10/30 (303)	10/31 (304)				

NOVEMBER						
11/1 (305)	11/2 (306)	11/3 (307)	11/4 (308)	11/5 (309)	11/6 (310)	11/7 (311)
11/8 (312)	11/9 (313)	11/10 (314)	11/11 (315)	11/12 (316)	11/13 (317)	11/14 (318)
11/15 (319)	11/16 (320)	11/17 (321)	11/18 (322)	11/19 (323)	11/20 (324)	11/21 (325)
11/22 (326)	11/23 (327)	11/24 (328)	11/25 (329)	11/26 (330)	11/27 (331)	11/28 (332)
11/29 (333)	11/30 (334)					

DECEMBER						
12/1 (335)	12/2 (336)	12/3 (337)	12/4 (338)	12/5 (339)	12/6 (340)	12/7 (341)
12/8 (342)	12/9 (343)	12/10 (344)	12/11 (345)	12/12 (346)	12/13 (347)	12/14 (348)
12/15 (349)	12/16 (350)	12/17 (351)	12/18 (352)	12/19 (353)	12/20 (354)	12/21 (355)
12/22 (356)	12/23 (357)	12/24 (358)	12/25 (359)	12/26 (360)	12/27 (361)	12/28 (362)
12/29 (363)	12/30 (364)	12/31 (365)				

Note: For leap year, add one to each number starting at 3/1 (60).

SECTION III
EXEC CALLS

INTRODUCTION

This section describes the basic formats of FORTRAN, FORTRAN IV, and Assembly Language EXEC calls with each call presented in detail. Table 3-1 is a summary of the EXEC calls listed in the order of appearance in this section. The error messages associated with the calls are listed at the end of this section. Refer to Appendix F at the rear of this manual for a summary of the EXEC calls and required parameters.

Table 3-1. RTE-C EXEC Calls

Call	Function
READ	To transfer information from an external I/O device into the system.
WRITE	To transfer information to an external I/O device from the system.
I/O CONTROL	To instigate various I/O control operations.
I/O STATUS	To request information about a device.
PROGRAM COMPLETION	To logically terminate a calling program.
PROGRAM SUSPEND	To suspend the calling program from execution.
PROGRAM SCHEDULE	To schedule a dormant program for execution.
TIME REQUEST	To request the current real-time.
EXECUTION TIME	To schedule a program for execution at specified time intervals, starting at a certain time.

An EXEC call is a block of words consisting of a "JSB EXEC" instruction and a list of parameters defining the request. The execution of the "JSB EXEC" instructions causes a memory protect violation interrupt and transfers control into RTE-C. RTE-C then determines the type of request (from the parameter list) and, if it is legally specified, initiates processing of the request.

In FORTRAN and FORTRAN IV, EXEC calls are coded as CALL statements. In Assembly Language, EXEC calls are coded as JSB EXEC 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.

ASSEMBLY LANGUAGE FORMAT

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

```
        EXT   EXEC      Used to link program to RTE-C
         .
         .
         .
        JSB   EXEC      Transfer control to RTE-C
        DEF   *+n+1     Defines point of return from
                        RTE-C; n is number of
                        parameters and 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      ---     )
```

FORTRAN/FORTRAN IV FORMAT

In FORTRAN and FORTRAN IV, 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 integer values or integer variables defined elsewhere in the program.

Example

```
CALL EXEC (7)
    or
ICODE = 7
CALL EXEC (ICODE)
```

Equivalent calling sequences

Some EXEC Call functions are handled automatically by the FORTRAN compilers or special subroutines. Refer to "FORTRAN," Section IV, Real-Time Program Preparation, and the specific EXEC calls.

READ/WRITE

Purpose:

To transfer information to or from an external I/O device. For a READ request, or if the I/O device is not buffered, the program is placed in the I/O suspend list until the operation is complete. RTE-C then reschedules the program.

Assembly Language:

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      Transfer control to RTE-C
            DEF   *+ 5, 6 or 7  Point of return from RTE-C; 6 or 7 is for additional
                            parameters
            DEF   ICODE     Request code
            DEF   ICNWD     Control information
            DEF   IBUFR     Buffer location
            DEF   IBUFL     Buffer length
            DEF   IPRM1     Optional parameter
            DEF   IPRM2     Optional parameter
            return point    Continue execution
             .
             .
ICODE       DEC   1 (or 2)  1=READ, 2=WRITE
ICNWD       OCT   conwd     conwd is described in Comments
IBUFR       BSS   n         Buffer of n words
IBUFL       DEC   n (or -2n)  Same n; words (+) or characters (-)
IPRM1       DEC   f         Optional parameter
IPRM2       DEC   q         Optional parameter
```

FORTRAN:

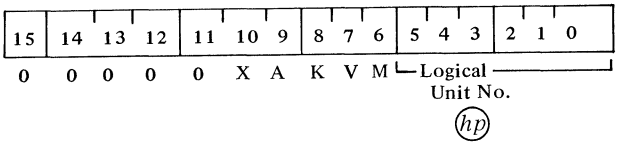
I/O transfers to most devices are programmed by standard READ and WRITE statements in FORTRAN.

COMMENTS

Parameters IPRM1 and IPRM2 are optional, except in the case of buffered devices where they cannot be used. In calls to I/O devices, these parameters may have many uses. For example, driver DVR77 (HP 2323A Subsystem) uses IPRM1 for the scanner channel number and IPRM2 for the instrument program word.

CONTROL WORD

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



(hp) *Modified to contain request code before entry into driver.*

Where:

M = 0 for ASCII.

M = 1 for binary.

V = 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 from the tape.

V = 0, and M = 1, the length of the punched tape input is determined by the buffer length specified in the EXEC call.

K = 1 causes keyboard input to be printed as received. If K = 0 input from the keyboard is not printed.

A = 1 designates punching ASCII characters on the teleprinter (M = 0). ASCII is usually printed; but since it is sometimes desirable to punch ASCII tapes, this option is provided. (If A = 0, M determines mode of transfer.)

X = When paper tape devices are used, "X" in combination with "M" and "V" will indicate an honesty mode that is defined as follows:

On input, if "X", "M", and "V" are set, absolute binary tape format is expected and handled. If "X" and "M" are set, and "V" is not, leader is not skipped and the specified number of words are read. On output, the record terminator (usually four feed frames) is not punched.

On input, if "X" is set and "M" is not, ASCII tape format is expected. Leader is not skipped, bit 8 is stripped, but otherwise, all characters are passed to the user's buffer. The only exception is line-feed, which terminates the record. On output, carriage return and line-feed are suppressed; any trailing left arrow is not (i.e., left arrow is transmitted but carriage return/line feed is not).

In an Assembly Language calling sequence, the buffer length can be a positive number for words (+) or a negative number for characters (-).

End-of-operation information is transmitted to the program in the A- and B-Registers. The A-Register contains word 5 (status word) of the device EQT entry with bits 14 and 15 indicating the end-of-operation status as defined by the driver completion code. This will be either 00-up, or 01-down. The B-Register contains a positive number which is the number of words or characters (depending upon which the program specified) actually transmitted.

NOTE

When a REAL array is transmitted, the buffer length must still be the total number of words required (i.e., 2 times REAL array length, or 3 times double precision array length).

I/O CONTROL

Purpose:

To carry out various I/O control operations, such as backspace, write end-of-file, rewind, etc. If the I/O device is not buffered, 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-C
        DEF   *+ 4 (or 3)   Point of return from RTE-C
```

(Continued on next page)

I/O Control (Continued)

```
                DEF   ICODE       Request code
                DEF   ICNWD       Control information
                DEF   IPRAM       Optional parameter
                return point      Continue execution
                 .
                 .
ICODE           DEC   3           Request code = 3
ICNWD           OCT   conwd       See Control Word
IPRAM           DEC   n           Required for some functions; see Control Word
```

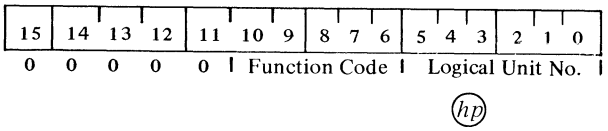
FORTRAN:

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

```
ICODE = 3              Request code
ICNWD = conwd          See Control Word
IPRAM = x              Optional; see Control Word
CALL EXEC (ICODE, ICNWD, IPRAM)
CALL EXEC (ICODE, ICNWD)
```

CONTROL WORD

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



(hp)

Function Code (Octal)	Action
00	Unused
01	Write end-of-field (magnetic tape)
02	Backspace one record (magnetic tape)
03	Forward space one record (magnetic tape)
04	Rewind (magnetic tape)
05	Rewind standby (magnetic tape)
06	Dynamic status (magnetic tape)
07	Set end-of-paper tape
10	Generate paper tape leader
11	List output line spacing
12	Write gap (magnetic tape)
13	Forward space file (magnetic tape)
14	Backward space file (magnetic tape)

(hp) *Modified to contain request code before entry into driver.*

The following functions are defined for DVR00:

20 Enable terminal – allows terminal to schedule its program when any key is struck.

21 Disable terminal – inhibits scheduling of terminal's program.

22 Set time-out – the optional third parameter is set as the new time-out interval.

23 Ignore all further action requests until:

a) The Queue is empty or
b) An input request is received or
c) A restore control request is received.

24 Restore output processing (this request is usually not needed).

Function Code (octal) 11 (list output line spacing), requires the optional parameter IPRAM which designates 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-C
            DEF     *+ 4 (or 5) Point of return from RTE-C
            DEF     ICODE       Request code
            DEF     ICNWD       Control information
            DEF     ISTA1       Status word 1
            DEF     ISTA2       Status word 2 - optional
            return point        Continue execution
             .
             .
ICODE       DEC     13          Request code = 13
ICNWD       DEC     n           Logical unit number
ISTA1       NOP                 Word 5 of EQT entry returned here
ISTA2       NOP                 Word 4 returned here, optional
```

FORTRAN:

```
            ICODE = 13          Request code
            ICNWD = nn          nn is the logical unit number
            CALL EXEC (ICODE, 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 V. Real-Time Input/Output.

PROGRAM COMPLETION

Purpose:

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

Assembly Language:

```
            EXT     EXEC
             .
             .
            JSB     EXEC        Transfer control to RTE-C
            DEF     *+ 2        Return point from RTE-C
            DEF     ICODE       Request code
            return point        Continue execution
             .
             .
ICODE       DEC     6           Request code = 6
```

FORTRAN:

The FORTRAN compiler generates a Program Completion EXEC Call automatically when it compiles an 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-C
        DEF   *+ 2          Point of return from RTE-C
        DEF   ICODE         Request code
        return point        Continue execution
         .
         .
         .
ICODE   DEC   7             Request code = 7
```

FORTRAN:

The FORTRAN library subroutine PAUSE, which is automatically called by a PAUSE statement, generates the Suspend EXEC Call.

COMMENTS

When a program is suspended, both the A- and B-Registers are saved. When the program is restarted with the GO request and no parameters, both registers are restored as they were at the point of suspension and the program continues.

When the program is restarted with the GO request **and** parameters, the B-Register contains the address of a five-word parameter array set by the GO request. In a FORTRAN program, the library subroutine RMPAR can load these parameters using the address in the B-Register as a pointer. It must be noted, however, that when RMPAR is used, parameters **must** accompany the GO request. Otherwise RMPAR uses the restored B-Register as an address to parameters which do not exist. The call to RMPAR must occur immediately following the Suspend EXEC Call, as in the following example:

```
DIMENSION I(5)      Must always specify at least 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 II.)

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-C
                DEF     *+ 3 + n    Return point, n = number of parameters
                DEF     ICODE       Request code
                DEF     INAME       Name of program to schedule
                DEF     IPRM1  )
                 .              )   Up to five optional parameters
                 .              )
                DEF     IPRM5  )
                return point        Continue execution
                 .
                 .
ICODE           DEC     9 or 10      9 = schedule with wait,
                                     10 = no wait; see Comments
INAME           ASC     3,xxxxx     xxxxx is the name of the program to schedule
IPRM1  )
       )                            Up to five optional parameters
IPRM5  )
```

FORTRAN:

Can only pass value parameters, not address.

```
DIMENSION INAME (3)
ICODE      = 9 (wait) or 10 (no wait)
INAME (1) = xxxxxB  )
INAME (2) = xxxxxB  )   Name of program in octal; see previous request,
                    )   Comments.
INAME (3) = xxxxxB  )
CALL EXEC (ICODE, INAME, IPRM1 ... IPRM5)
```

COMMENTS

If the program to be scheduled is dormant, it is scheduled and a zero is returned to the calling program in the A-Register.

If the program to be scheduled is not dormant, it is not scheduled by this call, and its status (which is some non-zero value) is returned to the calling program in the A-Register.

WAITING AND NO WAITING

A Schedule EXEC Call with waiting (ICODE = 9) causes RTE-C 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-C resume execution of the original program at the point immediately following the schedule call.

A Schedule EXEC Call with no waiting (ICODE = 10) 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). In FORTRAN, an immediate call to the library subroutine RMPAR retrieves the parameters from the scheduled request. For example:

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

The Program Schedule EXEC Call is similar to the ON operator request. (See Section II.) 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-C
        DEF   *+ 3      Point of return from RTE-C
        DEF   ICODE     Request code
        DEF   ITIME     Time value array
        return point    Continue execution
         .
         .
ICODE   DEC   11        Request code = 11
ITIME   BSS   5         Time value array
```

FORTRAN:

```
ICODE = 11
DIMENSION ITIME (5)
CALL EXEC (ICODE, ITIME)
```

COMMENTS

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

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

The Time Request EXEC Call is similar to the TI operator request. (See Section II.)

EXECUTION TIME (Initial Offset)

Purpose:

To schedule a program for execution at specified time intervals, starting after an initial offset time. RTE-C places the specified program in the time list and returns to the calling program.

Assembly Language:

```
                EXT     EXEC
                 .
                 .
                JSB     EXEC        Transfer control to RTE-C
                DEF     *+ 6        Point of return from RTE-C
                DEF     ICODE       Request code
                DEF     IPROG       Program to put in time list
                DEF     IRESL       Resolution code
                DEF     MTPLE       Execution multiple
                DEF     IOFST       Initial time offset
                return point        Continue execution
                 .
                 .
ICODE           DEC     12          Request code = 12
                DEC     0           Put calling program in time list
IPROG             or
                ASC     3,xxxxx     xxxxx is the program to put in the time list
IRESL           DEC     x           x is the resolution code (1=10's/ms; 2=secs; 3=mins;
                                    4=hrs)
MTPLE           DEC     y           y is the execution multiple
IOFST           DEC     -z          z (units set by x) gives the initial offset
```

FORTRAN:

```
IPROG = 0 or DIMENSION IPROG (3)
      IPROG (1) = xxxxxB  }
      IPROG (2) = xxxxxB  }   Define program name;
      IPROG (3) = xxxxxB  }
ICODE = 12
IRESL = x    (1=10's/ms; 2=secs; 3=mins; 4=hrs)
MTPLE = y
IOFST = -z     z (units set by x) gives the initial offset
CALL EXEC (ICODE, IPROG, IRESL, MTPLE, IOFST)
```

COMMENTS

The Execution Time EXEC Call is similar to the IT Operator request (see Section II). However, the EXEC Call places the program in the time list whereas IT does not. This call can schedule a program to execute in one of three ways as described in the following paragraphs.

RUN ONCE

After a time offset, the program will execute once and then be made dormant. This is accomplished as shown in the following example.

IRESL = 3 (specifies minutes)
MTPLE = 0 (specifies run once)
IOFST = -45 (specifies run after 45 minutes have elapsed from current time)

RUN REPEATEDLY

After a time offset, the program will execute, go dormant, and then re-execute at specified intervals. This is accomplished as shown in the following example.

IRESL = 3 (specifies minutes)
MTPLE = 60 (specifies run every 60 minutes)
IOFST = -30 (specifies run after 30 minutes have elapsed from current time)

GO DORMANT, THEN RUN

If IPROG=0, the current/calling program is made dormant, but the point of suspension is retained. The program is then placed in the time list for rescheduling from the point of suspension after a delay. When the program is rescheduled, it can be either to run once or repeatedly.

EXECUTION TIME (Absolute Start Time)

Purpose:

To schedule a program for execution at specified time intervals, starting at a particular absolute time. RTE-C places the specified program in the time list and returns to the calling program.

Assembly Language:

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      Transfer control to RTE-C
            DEF   *+9       Point of return from RTE-C
            DEF   ICODE     Request code
            DEF   IPROG     Program to put in time list
            DEF   IRESL     Resolution code
            DEF   MTPLE     Execution multiple
            DEF   IHRS      Hours
            DEF   MINS      Minutes
            DEF   ISECS     Seconds
            DEF   MSECS     Tens of milliseconds
            return point    Continue execution
             .
             .
             .
ICODE       DEC   12        Request code = 12
            DEC    0        Putting calling program in time list
IPROG         or
            ASC   3,xxxxx   xxxxx is the program to put in the time list
IRESL       DEC   x         x is the resolution code (1=10's/ms; 2=secs; 3=mins;
                            4=hrs)
MTPLE       DEC   y         y is the execution multiple
IHRS        DEC   a         Absolute starting time
MINS        DEC   b         In hours, minutes, seconds,
ISECS       DEC   c         and tens of milliseconds,
MSEC        DEC   d         on a 24-hour clock
```

FORTRAN:

```
IPROG = 0 or DIMENSION IPROG (3)
      IPROG (1) = xxxxxB
      IPROG (2) = xxxxxB          Define program name;
      IPROG (3) = xxxxxB
ICODE = 12
IRESL  = x        (1=10's/ms; 2=secs; 3=mins; 4=hrs)
MTPLE = y
IHRS  = h
MINS  = m
ISECS = s
MSECS = ms
CALL EXEC (ICODE, IPROG, IRESL, MTPLE, IHRS, MINS, ISECS, MSECS)
```

COMMENTS

The Execution Time EXEC call is similar to the IT Operator request (see Section II). However, the EXEC call places the program in the time list whereas IT does not. This call differs from the Initial Offset version in that a future starting time is specified instead of an offset. For example, if the current time is 1400 hours and you wish the program to be run at 1545 hours the parameters would be as follows:

IHRS = 15
MINS = 45
ISECS = 0
MSECS = 0

This call can schedule a program to execute in one of two ways as described in the following paragraphs.

RUN ONCE

At the absolute start-time, the program will execute once and then be made dormant. This is accomplished as shown in the following example.

IRESL = 3 (Specifies minutes)
MTPLE = 0 (specifies run once)
IHRS = *h*
MINS = *m* (specifies absolute start-time)
ISECS = *s*
MSECS = *ms*

RUN REPEATEDLY

At the absolute start-time, the program will execute, go dormant, and then re-execute at specified intervals. This is accomplished as shown in the following example.

IRESL = 3 (specifies minutes)
MTPLE = 60 (specifies run every 60 minutes)
IHRS = *h*
MINS = *m* (specifies absolute start-time)
ISECS = *s*
MSECS = *ms*

ERROR MESSAGES

When RTE-C, discovers an error in an EXEC call, it terminates the program, prints an error message on the operator console, and proceeds to execute the next program in the scheduled list.

When RTE-C aborts a program, it prints this message:

name ABORTED

When a memory protect violation occurs that is not an EXEC call, a resident library call, or $LIBX or $LIBR call, this message is printed: (*address* is the location that caused the violation.)

MP *name address*

When an EXEC Call contains an illegal request code, this message is printed: (*address* is the location that made the illegal call.)

RQ *name address*

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

SC01 = Missing parameter

SC02 = Illegal parameter

SC03 = Program cannot be scheduled

SC03 INT *name* occurs when an external interrupt attempts to schedule a program that is already scheduled. RTE-C ignores the interrupt and returns to the point of interruption.

name is the program name.

SC05 = Program given is not defined.

SC06 = No resolution code in Execution Time EXEC Call.

ERROR CODES FOR I/O CALLS

IO01 = Not enough parameters
IO02 = Illegal logical unit
IO03 = Logical unit not assigned
IO04 = Illegal user buffer
IO05 = Not used
IO06 = Not used
IO07 = Driver has rejected call
IO08 = Not used
IO09 = Not used

ERROR RETURN POINT

The user can alter the error return point of EXEC calls in association with error codes SC, IO, and RQ as shown in the following example.

```
CALL EXEC (ICODE ... )
GO TO error routine
normal return
```

This special error return is established by setting bit 15 to "1" on the request code word (ICODE). This causes the system to execute the first line of code following the CALL EXEC if there is an error, or if there is no error, the second line of code following the CALL EXEC.

The special error return will also return control to the calling program on a disc parity error on the system disc. In this case the B-Register will be set to –1 instead of the transmission log and the return will be to the normal return point. If there is an error the A-Register will be set to the ASCII error type (SC, IO, and RQ) and the B-Register set to the ASCII error numbers as usually printed on the system teletype.

The following excerpts from an example program demonstrates the use of the special error return.

```
FTN,L
     PROGRAM PROGA
     DIMENSION IREG(2)
     EQUIVALENCE (REG,IREG,IA),(IREG(2),IB)
          :
          :
```

When an EXEC call is issued, the sign bit must be set in the request code.

```
CALL EXEC (ICODE+1ØØØØØB,...)
GO TO 10   (ERROR RETURN POINT)
     .     (NO ERROR RETURN POINT)
     :
     :
```

After the following function is executed, "IA" will contain the A-Register contents and "IB" the B-Register contents. Note the EQUIVALENCE statement at the beginning of the example.

```
10 REG = AB(J)
```

The above function generates the following assembly code.

```
JSB AB     CALL ROUTINE AB.
DEF *+2    RETURN POINTER
DEF J      DUMMY ARGUMENT, MAKES AB
           EXTERNAL TO PROGRAM,
JSB .DST   STORE A AND B REGISTERS
DEF REG      IN REG
   .       CONTINUE
```

Next call the user defined error routine and pass it the error code.

```
CALL IER(IA,IB)
     :
     :
END
```

The following is a user written dummy routine for obtaining the A- and B-Registers.

```
ASMB,L
     NAM AB
     ENT AB
AB   NOP
     STA TMP
     LDA AB,I
     STA AB
     LDA TMP
     JMP AB,I
TEMP NOP
     END
```

SECTION IV
REAL-TIME PROGRAM PREPARATION

INTRODUCTION

This section is divided into two parts that describe the steps necessary to prepare, off-line, an Assembly or FORTRAN Language program for ultimate loading into the RTE-C System. Part 1 provides instructions for Assembly Language programs, and Part 2 for FORTRAN programs. Once the source program is assembled or compiled, and a relocatable binary tape is obtained, the relocatable tape is either configured directly into RTE-C during RTSGN time, or configured into absolute by the Relocating Loader for later on-line loading into the system with the Absolute Program Loader (APLDR).

The Assembler and Compiler require the use of SIO Drivers for input/output. More complete and detailed information can be found in the appropriate manuals as listed:

HP Assembler	02116-9014
HP FORTRAN	02116-9015
Software Operating Procedures	
Software Input/Output System Configuration	5951-1374
SIO Subsystem Operation	5951-1390

Part I
Assembly Language Program Preparation

ASSEMBLY PROCESSING

The RTE-C System uses a modified version of the HP Extended Assembly Lanugage to permit parameters in the NAM statement. The Assembler (SIO) is a two-pass system, or, if both punch and list output are requested, a three-pass system on a minimum configuration. A pass is defined as a processing cycle of the source program input.

In the first pass, the Assembler creates a symbol table from the names used in the source statements. It also checks for certain possible error conditions and generates diagnostic messages if necessary.

During pass two, the Assembler again examines each statement in the source program along with the symbol table and produces the binary program and a program listing. Additional diagnostic messages may also be produced.

If only one output device is available, and if both the binary output and the list output are requested, the listing function is deferred and performed as pass three.

When using the Assembler with a mass storage device, the source program is written on the device during the first pass; the second pass of the source is read from the mass storage.

ASSEMBLY FORMAT

Refer to the HP Assembler Manual (02116-9014) for the form of the control statement. Normally, only relocatable code is generated for RTE-C. The relocatable code can be configured into the system during generation time, or it can be processed by the Relocating Loader which produces an absolute program for on-line loading with the Absolute Program Loader (APLDR).

NAM STATEMENT

Purpose:

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

(Continued)

NAM STATEMENT (Continued)

Format:

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 0 if not given):

0 = System program
1 = Real-Time core-resident
2 = Not Used
3 = Not Used
4 = Not Used
5 = Not Used
6 = Library (re-entrant or privileged)
7 = Utility

p2 is the priority (1 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 0 if not given. See Section II, IT, for meaning)

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

COMMENTS

The NAM statement may include optional parameters defining the program type, priority, and time values if the program was compiled with the RTE (Disc Based) Assembler. The parameters can also be entered (or changed) during RTSGN time after the relocatable program is loaded and the generator requests the operator to ENTER PRAMS. Refer to the RTE-C System Installation, Section VI, for more details. The NAM statement format in the above box is an example of the parameters referred to.

ORB STATEMENT

The ORB statement can be used in the RTE-C System to store data (e.g., constants) on the base page for easy-access, read-only processing. An example might be:

```
          .
          .
          .
          ORB
B5        OCT 5
B6        OCT 6
Bn        OCT X
          .
          .
          .
          ORR
          LDA B5
          STA BUFF
```

Where the constants B5 through B*n* are stored on the base page and are accessed by the label. Note that 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-C when an interrupt occurs, and RTE-C checks whether it was an EXEC call

(JSB EXEC, JSB $LIBR, JSB $LIBX).

If not, the user program is aborted.

OPERATING INSTRUCTIONS

The instructions presented here are abbreviated from the detailed instructions located in the Software Operating Procedures Manual, Software Input/Output System Configuration (5951-1374), and SIO Subsystem Operation (5951-1390).

Load the SIO Configured Assembler tape into memory using the Basic Binary Loader (BBL).

Load the source tape using one of the following starting addresses:

Octal 100 when ASMB Control Statement is on tape.
Octal 120 when ASMB Control Statement is entered through terminal keyboard.

Replace the source tape in the input device and select desired switch register options – press RUN.

Part II
Fortran Language Program Preparation

FORTRAN PROCESSING

The RTE-C System uses a modified version of the standard HP FORTRAN Compiler. The modification is to the control statement and causes the correct EXEC calling sequence to be generated. Refer to the FORTRAN Control Statement later in this part for more information. Programs that have been compiled with the RTE (Disc Based) FORTRAN or FORTRAN IV Compilers can be loaded and run on the RTE-C System.

The Compiler is a two-pass system where the source tape is compiled with FORTRAN Pass 1 resulting in an intermediate tape. FORTRAN Pass 2 then compiles this intermediate tape and produces the relocatable binary object tape.

FORTRAN FORMAT

The RTE-C FORTRAN Language is similar to the regular HP FORTRAN Language (refer to HP FORTRAN Manual Part No. 02116-9015). The differences are described in the next few paragraphs. RTE-C FORTRAN has additional capabilities, using EXEC calls. Read Section III for complete details on the EXEC calls.

FORTRAN CONTROL STATEMENT

Purpose:

To define the output to be produced by the FORTRAN Compiler.

Format:

FTN,B,L,A

Where:

B = Binary Output

L = List output

A = Assembly listing

Besides the standard options shown above, two additional compiler options. T and *n*, are available.

(Continued)

FORTRAN CONTROL STATEMENT (Continued)

T

Lists the symbol table for each program in the compilation. The A option includes the 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, ERRO, is used. The error routine is called when an error occurs in ALOG, SQRT, .RTOR, SIN, COS, .RTOI, EXP, .ITOI or TAN.

PROGRAM STATEMENT

Purpose:

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

Format:

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)

0 = System Program
1 = Real-Time Core-Resident
2 = Reserved
3 = Reserved
4 = Reserved
5 = Reserved
6 = Library (re-entrant or privileged)
7 = Utility

(Continued on next page)

PROGRAM STATEMENT (Continued)

p2	is the priority (1 to 99, set to 99 if not given)	
p3	is the resolution code	
p4	is the execution multiple	
p5	is hours	(Time values are set to 0 if not given. See Section II, IT, for meaning)
p6	is minutes	
p7	is seconds	
p8	is tens of milliseconds	

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

COMMENTS

The Program statement may include optional parameters defining the program type, priority, and time values. These parameters may be changed (or entered if none were specified) during RTSGN time after the relocatable program is loaded and the generator requests the operator to ENTR PRAMS. Refer to the RTE-C System Installation Section VI for more details. The PROGRAM statement format in the following box is an example of the parameters referred to.

PAUSE & STOP STATEMENTS

Purpose:

PAUSE provides a temporary program halt.

Format:

name: PAUSE *xxxx*

The program is then suspended (see EXEC Suspend Call).

Where:

name is the program name, and

xxxx HP is the octal number given in the PAUSE.

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

(Continued)

PAUSE & STOP STATEMENTS (Continued)

Purpose:

STOP causes the program to be terminated.

Format:

name: STOP *xxxx*

Where:

name is the program name, and

xxxx is the octal number given in STOP. *(Does not require the 'B' octal designator suffix.)*

ERR0 LIBRARY ROUTINE

Purpose:

Prints the following message whenever an error occurs in a library routine:

Format:

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 ERR0 automatically.

If the FORTRAN control statement includes an *n* option, the call will be to ERR*n*, a routine which the user must supply.

Read the FORTRAN manual for the meaning of error codes.

OPERATING INSTRUCTIONS

The instructions presented here are abbreviated from the detailed instructions located in the Software Operating Procedures Manual, Software Input/Output System Configuration (5951-1374), and SIO Subsystem Operation (5951-1390).

Load the SIO configured FORTRAN Pass 1 tape into memory with the Basic Binary Loader (BBL).

Load the source tape using one of the following addresses:

Octal 100 when FTN Control Statement is on tape.

Octal 120 when FTN Control Statement is entered through terminal keyboard.

Load the FORTRAN Pass 2 tape into memory with the BBL.

Load the intermediate tape that was punched from the Pass 1 operation using the starting address of octal 100.

SECTION V
REAL-TIME INPUT/OUTPUT

INTRODUCTION

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-C relieves the programmer of most I/O problems.

For further details on the hardware input/output organization, consult the appropriate computer manuals.

SOFTWARE I/O STRUCTURE

An Equipment Table records each device's I/O channels, driver, DMA, buffering and time-out specifications. A Device Reference Table assigns one or more logical unit numbers to each entry in the Equipment Table, thus allowing the programmer to reference changeable logical units instead of fixed physical units.

An Interrupt Table directs the system's action when an interrupt occurs on any channel; RTE-C can call a driver (which is responsible for initiating and continuing operations on all devices of an equivalent type), schedule a specified program, or handle the interrupt itself.

The programmer requests I/O by means of an EXEC Call in which he specifies the logical unit, control information, buffer location, buffer length, and type of operation. Other devices or subsystems may require additional parameters.

THE EQUIPMENT TABLE

The Equipment Table (EQT) has an entry for each device recognized by RTE-C (these entries are established by the user when the RTE-C System is generated). These 15-word EQT entries reside in the system, and have format shown in Figure 5-1.

Table 5-1. Equipment Table

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			S	T					Unit #			Channel #		
5	AV		EQUIP TYPE CODE								STATUS					
6								Function Code						Request Code		
7	Request Buffer Address															
8	Request Buffer Length															
9	Temporary Storage for Optional Parameter															
10	Temporary Storage for Optional Parameter															
11	Temporary Storage for Driver															
12	Temporary Storage for Driver															
13	Temporary Storage for Driver															
14	Device Time-Out Reset Value															
15	Device Time-Out Clock															

Where:

D = 1 if DMA required.
B = 1 if automatic output buffering used.
S = 1 if driver is to process Time-out.
T = 1 if device timed out (system sets to zero before each I/O request).
Unit = Last sub-channel addressed.
Channel = I/O select code for device (lower number if a multi-board interface).
AV = availability indicator:
0 = 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.

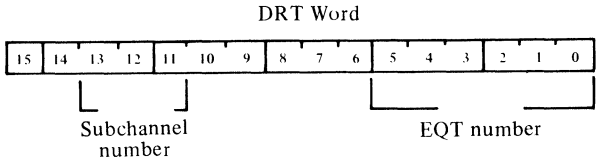
EQUIP. TYPE CODE = type of device. When this octal number is linked with "DVR", it identifies the device's software driver routine, for example:

00 to 07 = paper tape devices (or system control devices)
00 = teleprinter (or system keyboard control device)
01 = photoreader
02 = paper tape punch
10 to 17 = unit record devices
10 = plotter
12 = line printer
20 to 37 = magnetic tape/mass storage devices
40 to 77 = instruments
EXEC Call (See Section III) with Request Code in bits 0-5 and function code as in CONWD.

When RTE-C 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 B) before calling the driver routine.

LOGICAL UNIT NUMBERS

Logical unit numbers from decimal 1 to 63 provide logical addressing of the physical devices defined in the EQT and the subchannels within the physical devices (if applicable). These numbers are maintained in the Device Reference Table (DRT), which is created by RTSGN, 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, and the subchannel number within the EQT entry. The functions of logical units 1 through 6 are predefined in the RTE-C System as:

1 - system teleprinter
2 - reserved for system
3 - reserved for system
4 - standard punch unit
5 - standard input unit
6 - standard list unit

Logical units 7 through 63 may be assigned for any functions desired. The operator can assign EQT numbers and subchannel numbers within the EQT entries to the logical unit numbers when the RTE-C System is generated (see Section VI), or after the system is running (see Section II, LU). The user determines the number of logical units when the system is generated.

Logical unit numbers are used by executing programs to specify on which device I/O transfers are to 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 and subchannel 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 0, 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-C 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-C 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.

GENERAL OPERATION OF I/O PROCESSOR

A user program makes an EXEC Call to initiate I/O transfers. If the device is not buffered, or in the case of all

input transfers, the calling user program is suspended until the transmission is completed. The next lower priority program is allocated execution time during the suspension of a higher priority program.

An I/O request (i.e., Read, Write, Control) is channeled to IOC by the executive request processor. After the necessary legality checks are made, the request is linked into the suspension list corresponding to the referenced I/O device. The parameters from the request are set in the temporary storage area of the Equipment Table.

If the device is available (i.e., no prior requests were stacked), the "initiation" section of the associated driver is called. The initiation section initializes the device and starts the data transfer or control function. On return from the initiation section, or if the device is busy, or a required DMA channel is not available, IOC returns to the scheduling module to execute the next lower priority program.

Interrupts from the device causes the Central Interrupt Control (CIC) module to call the "completion" section of the driver. At the end of the operation, the driver returns to CIC and consequently to IOC. IOC causes the requesting program to be placed back into the schedule list and checks for a buffered I/O stacked request. If there are no stacked requests, IOC exits to the scheduling module (SCHED); otherwise, the initiation section is called to begin the next operation before returning.

DRIVER STRUCTURE AND OPERATION

An I/O driver, operating under control of the Input/Output Control (RTIOC) and Central Interrupt Control (CIC) modules of RTE-C, 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. It should be noted that RTE-C operation makes it mandatory that a synchronous device driver must use a DMA or privileged interrupt channel for data transfer.

An I/O driver always has an initiation section and usually a 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 respectively, and DVR*nn* is the driver name. Privileged drivers are in a special class. Refer to the end of this section for a discussion of privileged drivers.

INITIATION SECTION

The RTIOC module of RTE-C calls for initiation section directly when an I/O transfer is initiated. Locations EQT1 through EQT15 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 0 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-C places the request parameters from the user program's EXEC Call into words 6 through 10 of the EQT entry. The subchannel number is placed into bits 6 through 8 of word 4. Word 6, CONWD, is modified to contain the request code in bits 0 through 5 in place of the logical unit. See the EQT entry diagram in Table 5-1, and Section III, Read/Write Exec Call, for details of the parameters.

Once initiated, the driver can use words 6 through 13 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 discussion of special conditions under "Privileged Interrupt Processing").

The initiation section of the driver is responsible for these functions (as flow-charted in Figure 5-1).

a. Rejects the request and proceeds to "f" if:

1. The device is inoperable,
2. the request code, or other of the parameters, is illegal.

b. Configures all I/O instructions in the driver to include the select code (and DMA channel) of the device.

c. Initializes DMA, if appropriate.

d. Initializes software flags and activates the device. All variable information pertiment 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.

e. Optionally sets the device time out clock (EQT 15).

f. Returns to RTIOC with the A-Register set to indicate initiation or rejection and the cause of the reject:

If A = 0, then operation was initiated.

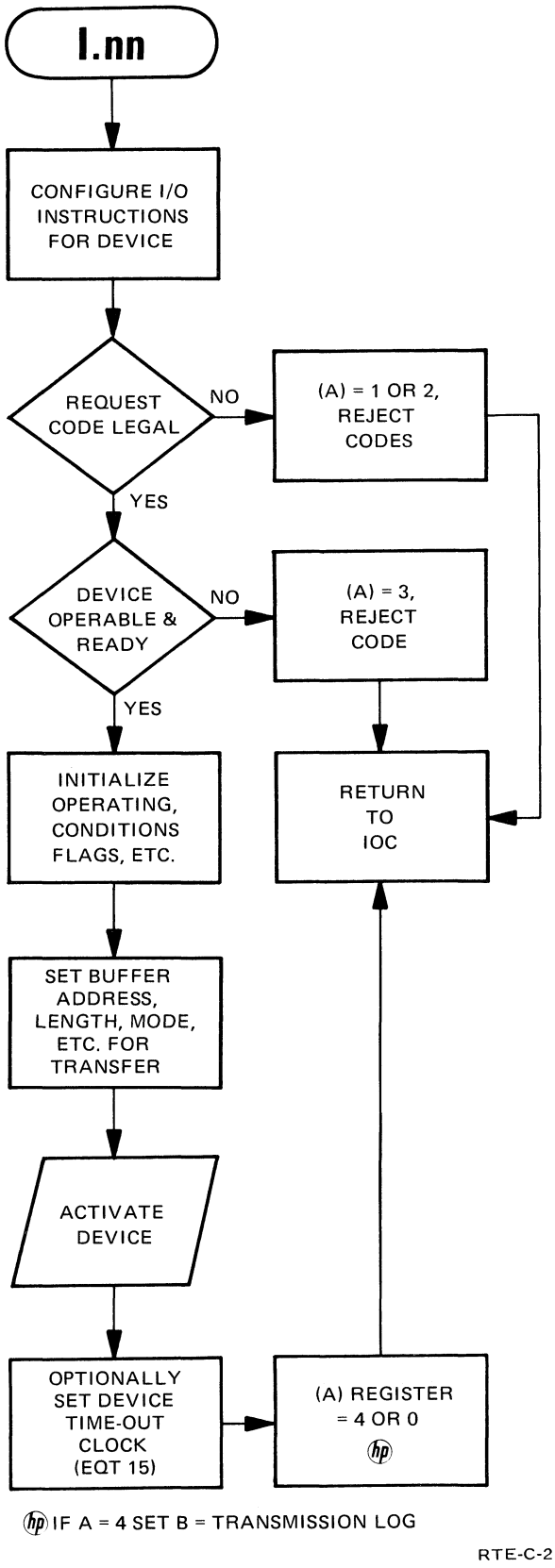


Figure 5-1. I/O Driver Initiation Section

If A = 1, 2, 3, then operation rejected because:
1 – read or write illegal for device,
2 – control request illegal or undefined,
3 – equipment malfunction or not ready,

If A = 4, immediate completion. (Transmission log should be returned in the B-Register in this case.)

DMA INITIALIZATION – A driver can obtain a DMA channel in two ways:

The channel can be assigned during generation by entering a "D" in the driver's Equipment Table Entry.

The driver can dynamically assign a DMA channel as required.

If a driver requires DMA but does not require or use the DMA interrupt, the DMA control should be cleared after DMA initialization. Further special processing is not required in this case.

If a driver requires DMA, and the DMA interrupt, 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 priviliged interrupt processing.)

The following code demonstrates these principles:

```
CLF 0                   Disable interrupts.

STC DMA, C              Initiate DMA

CLA                     Bypass this section if
CPA DUMMY               DUMMY = 0 and special
JMP X                   processing is not needed.

CLC DMA                 Clear DMA control. Set B =
LDB INTBA               address of the appropriate
LDA CHAN                entry in the interrupt table.
CPA = D7
INB
```

```
LDA B,I
IOR = B100000
STA B, I
STF 0
```

Set bit 15 of the entry equal to 1 and return to the interrupt table. Enable interrupt system.

X continue

There may be times when a driver will only occasionally need DMA, and thus not want to always tie up a DMA channel while it is operating: this may be done in one of two ways: (Note that in example No. 1, the DMA channel is always assigned before the driver is entered. In example No. 2, the DMA channel is assigned only if the driver requests it.

Example 1 – The DMA flag is set at RTSGN time by entering a "D" in the driver's equipment table entry. The driver may return the DMA channel (before completion if desired) by clearing the appropriate INTBL word (first or second word of interrupt table). This may be done as follows:

```
LDA    DMACH    GET CURRENT CHANNEL
LDB    INTBA    AND INTBL ADDRESS
SLA             IF CHANNEL 7
INB             STEP ADDRESS
CLA             CLEAR THE
STA    B,I      CHANNEL WORD
```

Example 2 – The DMA flag is not set at RTSGN time as above. In this case the driver is entered by RTIOC without a channel being assigned. The driver must analyze the request and determine if a channel is required, and if so, request a channel from RTIOC by returning via I.XX,I with A=5. RTIOC will assign a channel and recall the driver. The recall completely resets EQT words 6 through 10. Since it is possible for the calling program to be aborted between the request for DMA and the resulting recall of the driver, the driver must determine, independently of its past history, if it has DMA. The following code illustrates these principles:

```
.
.
.
DLD    INTBA,I  COME HERE IF DMA
                REQUIRED
CPA    EQT1     IS CHANNEL 6 ASSIGNED?
JMP    CH6      YES; GO CONFIGURE
CPB    EQT1     IS CHANNEL 7 ASSIGNED?
JMP    CH7      YES? GO CONFIGURE
LDA    =B5      NO CHANNEL SO
JMP    I.XX,I   REQUEST ONE FROM RTIOC
.
.
.
```

In this case the driver must also tell RTIOC that it has a DMA channel at completion of request. This is done by setting the sign bit in the A-Register on the completion return to RTIOC. This bit may be set at all times – even when the driver does not own a DMA channel. However, if set when not required, some extra overhead in RTIOC is incurred. The sign bit is set in addition to the normal completion code. The following code illustrates this principle:

```
.
.
.
LDA    COMCD    GET COMPLETION CODE
IOR    =B100000 SET THE SIGN BIT
JMP    C.XX,I   RETURN TO RTIOC
```

NOTE

If your driver wishes to do a series of non-DMA operations, but still retain the DMA channel assignment, you must clear bit 15 in the first or second word of the INTBL entry to prevent the system from restoring DMA. The correct word must be determined by the driver and is the word described in the above paragraphs. That is;

INTBL (1) - channel 1 (location 6)

INTBL (2) - channel 2 (location 7)

Programming Hint – A driver may use the following code to determine which DMA channel it is using at any time:

```
DLD INTBA, I    GET DMA WORDS
RAL,CLE,ERA     CLEAR SIGN
RBL,CLE,ERB     BITS (NEEDED ONLY IF
                DRIVER SETS THE SIGN BIT)
CPA EQT1        CHANNEL 6?
JMP CH6         YES
CPB EQT1        CHANNEL 7?
JMP CH7         YES
JMP NODMA       NO–NO DMA ASSIGNED
```

COMPLETION SECTION

RTE-C 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 disabled (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 flowcharted in Figure 5-2 and performs the following functions in the order indicated.

a. 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, sets EQT 15 (time-out clock) to zero to prevent time-out, 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 "b".

b. 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.

c. 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 "f".

d. 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 the EQT. The return to CIC must be (P+2) as in "f".

e. 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):

1. Set the actual or simulated device status, into bits 0 through 7 of EQT word 5.

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

3. Set the A-Register to indicate successful or unsuccessful completion and the reason:

A equals 0 for successful operation,
A does not equal 0 for unsuccessful:
1 – device malfunction or not ready,
2 – end-of-tape (information),
3 – transmission parity error.
4 – device Time-out

f. Clears the device and DMA control, if end-of-operation, or sets the device and DMA for the next transfer or retry. If not end-of-operation (i.e., a continuation exit is to be made), the driver can again optionally set the device time-out clock. Returns to CIC at:

(P+1) – completion with the A and B-Registers set as in "e".

(P+2) – continuation; the registers are not significant.

I/O DEVICE TIME-OUT

Each I/O device can have a time-out clock that will prevent indefinite I/O suspension. Indefinite I/O suspension can occur when a program initiates I/O, and the device fails to return a flag (possible hardware malfunction or improper program encoding). Without the device time-out, the program which made the I/O call would remain in I/O suspension indefinitely awaiting the operation-done indication from the device. With respect to privileged drivers, the time-out parameter must be long enough to cover the period from I/O initiation to transfer completion.

Two words, EQT 14 and EQT 15, of the EQT entry for each I/O device function as a device time-out clock. EQT 15 is the actual working clock, and before each I/O transfer is initiated, is set to a value *m*, where *m* is the two's complement of a number of 10 ms time intervals. If the device does not interrupt within the required time interval, it is to be considered as having "timed out". The EQT 15 clock word for each device can be individually set by two methods.

- The system inserts the contents of EQT 14 into EQT 15 before a driver (initiation or completion section) is entered. EQT 14 can be preset to *m* by entering (T=) *m* during the EQT entry phase of RTSGN execution (see RTSGN, Section VI), or it can be set or modified on-line with the TO operator request (see Section II).

- When the driver initiates I/O, and expects to be entered due to a subsequent interrupt, the driver can set the value *m* into EQT 15 just before it exits. This value *m* can be coded permanently into the driver or else passed to the driver as an I/O call parameter.

NOTE

The system always inserts the contents of EQT14 into EQT15 before entering a driver. However, a time-out value inserted directly into EQT15 by the driver over-rides any value previously set by the system (from EQT14).

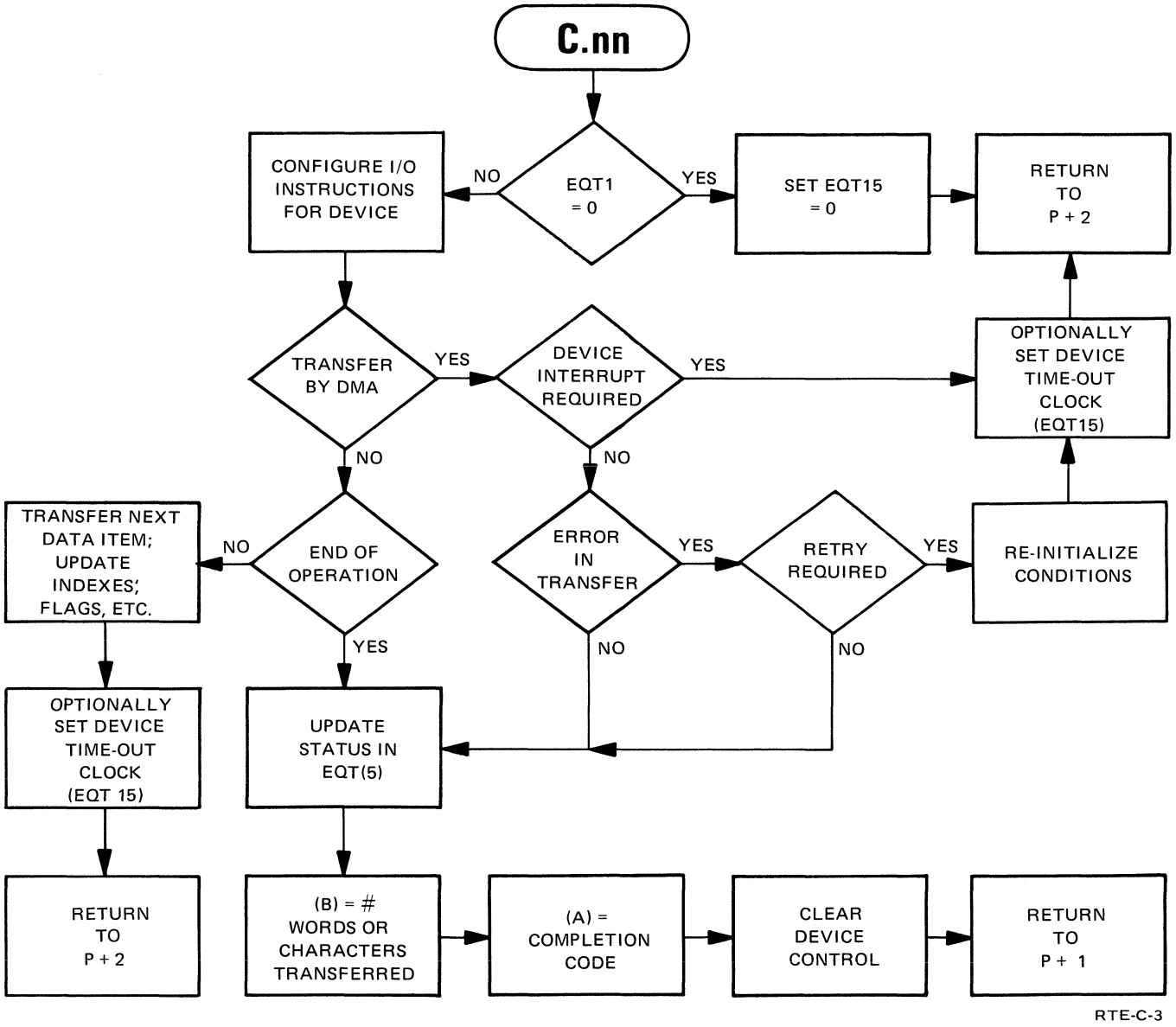


Figure 5-2. I/O Driver Completion Section

DRIVER PROCESSING OF TIME-OUT

A driver indicates to the system that it wants to process time-out by setting bit 12 in EQT word 4. The system never clears this bit so it need be set only once. In this case, when a device times out, the following actions take place:

a. Bit 11 in EQT word 4 is set.

b. The driver is entered at C.*nn* with the A-Register set to the select code (from EQT word 4).

c. The driver must recognize that the entry is for time-out by examining bit 11 of EQT word 4 and do whatever is necessary. The driver should then clear bit 11 in the event it is entered again prior to completion of the operation so that it knows why it is being entered on the next call. (RTIOC will clear this bit prior to entering the driver at I.*nn*.)

d. The driver may continue or complete the operation. If it completes the operation it may set the A-Register to 4 to indicate time-out.

e. If the A-Register is set to 4, RTIOC will issue the message

I/O ERROR TO EQT #*x*

where *x* is the EQT number and will set the device down.

SYSTEM PROCESSING OF TIME-OUT

In the case where the driver does not set bit 12 of EQT word 4, the following actions take place on time-out.

a. The calling program is rescheduled, and a zero transmission log is returned to it.

b. The device is set to the down status, and bit 11 in the fourth word of the device's EQT entry is set to one. An error message is printed; e.g.,

I/O ERROR TO EQT #*x*

c. The system issues a CLC to the device's select code(s) through the EQT number located in the interrupt table.

Due to the system issuing a CLC to the device's select code, each device interface card requires an entry in the interrupt table during RTSGN. If an I/O card did not normally interrupt, and therefore did not have an entry in the interrupt table, and the device had timed out, the system would not be able to issue a CLC to the I/O card. Without the CLC, the operator might not be able to reactuate the device with the UP operator request.

A time-out value of zero is equivalent to not using the time-out feature for that particular device. If a time-out parameter is not entered, its value remains zero and time-out is disabled for the device.

SAMPLE I/O DRIVER

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

```
PAGE 0002 HP1   ** RT EXEC DRIVER <70> G.P.R. (OUTPUT) **


0001                    ASMB,R,L
0003  000000                                NAM DVR70
0004*
0005*
0006                                        ENT I.70,C.70
0007*
0008*
0009*   DRIVER 70 OPERATES UNDER THE CONTROL OF THE
0010* I/O CONTROL MODULE OF THE REAL-TIME EXECUTIVE.
0011* THIS DRIVER IS RESPONSIBLE FOR CONTROLLING
0012* OUTPUT TRANSMISSION TO A 16 BIT EXTERNAL
0013* DEVICE. <70> IS THE EQUIPMENT TYPE CODE ASSIGNED
0014* GENERALLY TO THESE DEVICES.  I.70 IS THE
0015* ENTRY POINT FOR THE *INITIATION* SECTION AND C.70
0016* IS THE *COMPLETION* SECTION ENTRY.
```

```
0017*
0018*
0019* - THE INITIATION SECTION IS CALLED FROM I/O
0020*   CONTROL TO INITIALIZE A DEVICE AND INITIATE
0021*   AN OUTPUT OPERATION.
0022*
0023*   I/O CONTROL SETS THE ADDRESS OF EACH WORD OF THE
0024*   15 WORD EQT ENTRY (FOR THE DEVICE) IN THE SYSTEM
0025*   COMMUNICATIONS AREA FOR BOTH INITIATOR AND CONTINUATOR
0026*   SECTIONS.  THE DRIVER REFERENCES TO THE EQT ARE:
0027*   -EQT1 THRU EQT15-
0028*
0029*   CALLING SEQUENCE:
0030*
0031*        (A) = SELECT CODE OF THE I/O DEVICE.
0032*        P   JSB I,70
0033*        P+1 -RETURN-
0034*
0035*
0036*           (A) = 0, OPERATION INITIATED
0037*           (A) = REJECT CODE
0038*
0039*               1, ILLEGAL REQUEST
0040*               2, ILLEGAL MODE
0041*
0042* -THE COMPLETION SECTION IS CALLED BY CENTRAL
0043*   INTERRUPT CONTROL TO CONTINUE OR COMPLETE
0044*   AN OPERATION.
0045*
0046*     CALLING SEQUENCE:
0047*
0048*        (A) = SELECT CODE OF THE I/O DEVICE.
0049*
0050*        P   JSB C,70
0051*        P+1 -COMPLETION RETURN-
0052*        P+2   CONTINUATION RETURN-
0053*
0054*          (A) = 0, SUCCESSFUL COMPLETION WITH
0055*                (B) = # OF WORDS TRANSMITTED.
0056*          (A) = 2, TRANSMISSION ERROR DETECTED
0057*                (B), SAME AS FOR (A)=0


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


0058*
0059*          - CONTINUATION RETURN: REGISTERS
0060*              MEANINGLESS
0061*
0062* -RECORD FORMAT-
0063*
0064*    THIS DRIVER PROVIDES A 16 BIT BINARY
0065*     WORD TRANSFER ONLY.
```

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


0067*     *** INITIATION SECTION ***
0068*
0069  00000 000000  I,70  NOP            ENTRY FROM IOC
0070*
0071  00001 016071R       JSB SETIO      SET I/O INSTRUCTIONS FOR DEVICE
0072*
0073  00002 161665        LDA EQT6,I     GET CONTROL WORD OF REQUEST,
0074  00003 012105R       AND =B3         ISOLATE,
0075*
0076  00004 052106R       CPA =B1        IF REQUEST IS FOR INPUT
0077  00005 126000R       JMP I,70,I      THEN REJECT,
0078  00006 052107R       CPA =B2        PROCESS FOR WRITE REQUEST
0079  00007 026012R       JMP D,X1        GO TO WRITE REQUEST
0080*
0081* REQUEST ERROR- CAUSE REJECT RETURN TO I/O CONTROL,
0082*
0083  00010 062107R       LDA =B2        SET A=2 FOR ILLEGAL CONTRL REQ,
0084  00011 126000R       JMP I,70,I     -EXIT-
0085*
0086* WRITE REQUEST PROCESSING
0087*
0088  00012 161666  D,X1  LDA EQT7,I     GET REQUEST BUFFER ADDRESS
0089  00013 171670        STA EQT9,I      AND SET AS CURRENT ADDRESS
0090  00014 161667        LDA EQT8,I     GET BUFFER LENGTH
0091  00015 003004        CMA,INA         SET NEGATIVE AND SAVE
0092  00016 171671        STA EQT10,I     AS CURRENT BUFFER LENGTH,
0093  00017 002002        SZA            CHECK LENGTH
0094  00020 026024R       JMP D,X3       NON-ZERO
0095  00021 062110R       LDA =B4        IMMEDIATE COMPLETION
0096  00022 006400        CLB            SET TLOG IN B-REG
0097  00023 126000R       JMP I,70,I     IF ZERO
0098*
0099* CALL COMPLETION SECTION TO WRITE FIRST WORD,
0100*
0101  00024 062104R D,X3  LDA P2         ADJUST RETURN
0102  00025 072031R       STA C,70        TO INITIATOR SECTION,
0103  00026 026036R       JMP D,X2       GO TO COMPLETION SECTION
0104*
0105  00027 002400  IEXIT CLA            RETURN TO I/O CONTROL WITH
0106  00030 126000R       JMP I,70,I      OPERATION INITIATED,
0107*
```

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


0109*
0110*      *** COMPLETION SECTION ***
0111*
0112   00031 000000  C.70   NOP             ENTRY
0113   00032 165660         LDB EQT1,I      SPURIOUS
0114   00033 006003         SZB,RSS          INTERRUPT?
0115   00034 026051R        JMP SPURI       YES - IGNORE
0116   00035 016071R        JSB SETIO       SET I/O INSTRUCTIONS FOR DEVICE
0117*
0118   00036 002400  D.X2   CLA             IF CURRENT BUFFER LENGTH = 0,
0119   00037 151671         CPA EQT10,I      THEN,GO TO
0120   00040 026054R        JMP I.3          STATUS SECTION.
0121*
0122   00041 165670         LDB EQT9,I      GET CURRENT BUFFER ADDRESS
0123   00042 135670         ISZ EQT9,I      ADD 1 FOR NEXT WORD
0124   00043 160001         LDA B,I         GET WORD
0125   00044 135671         ISZ EQT10,I     AND INDEX WORD COUNT
0126   00045 000000         NOP             IGNORE P+1 IF LAST WORD.
0127*
0128*
0129   00046 102600  I.1    OTA 0           OUTPUT WORD TO INTERFACE
0130   00047 103700  I.2    STC 0,C         TURN DEVICE ON
0131   00050 002001         RSS
0132   00051 175774  SPURI  STB EQT15,I     ZERO TIME-OUT CLOCK WORD
0133*
0134   00052 036031R        ISZ C.70        ADJUST RETURN TO P+2
0135   00053 126031R        JMP C.70,I      -EXIT-
0136*
```

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


0138*
0139* STATUS AND COMPLETION SECTION.
0140*
0141  00054 102500  I.3   LIA 0          GET STATUS WORD
0142  00055 012111R       AND =B77       STRIP OFF BITS
0143  00056 070001        STA B           AND SAVE IN B
0144  00057 161664        LDA EQT5,I     REMOVE PREVIOUS
0145  00060 012112R       AND =B177400    STATUS BITS
0146  00061 030001        IOR B          SET NEW
0147  00062 171664        STA EQT5,I      STATUS BITS
0148*
0149  00063 002400        CLA            SET NORMAL RETURN COND
0150  00064 056110R       CPB =B4        ERROR STATUS BIT ON?
0151  00065 062107R       LDA =B2        YES, SET ERROR RETURN
0152*
0153  00066 165667        LDB EQT8,I     SET (B) = TRANSMISSION LOG
0154*
0155  00067 106700  I.4   CLC 0          CLEAR DEVICE
0156*
0157  00070 126031R       JMP C.70,I     *EXIT FOR COMPLETION
0158*
0159*
0160* SUBROUTINE <SETIO> CONFIGURES I/O INSTRUCTIONS.
0161*
0162  00071 000000  SETIO NOP
0163  00072 032103R       IOR LIA        COMBINE LIA WITH I/O
0164  00073 072054R       STA I.3        SELECT CODE AND SET.
0165*
0166  00074 042113R       ADA =B100      CONSTRUCT OTA INSTRUCTION
0167  00075 072046R       STA I.1
0168*
0169  00076 042114R       ADA =B1100     CONSTRUCT STC,C INSTRUCTION
0170  00077 072047R       STA I.2
0171*
0172  00100 032115R       IOR =B4000     CONSTRUCT CLC INSTRUCTION
0173  00101 072067R       STA I.4
0174*
0175  00102 126071R       JMP SETIO,I    *RETURN*
0176*
```

```
 PAGE 0007 #31  < DRIVER 70 *COMPLETION SECTION* >


0178*
0179* CONSTANT AND STORAGE AREA
0180*
0181  00000           A      EQU 0           A-REGISTER
0182  00001           B      EQU 1           B-REGISTER
0183*
0184  00103 102500   LIA    LIA 0
0185  00104 000026R P2     DEF IEXIT-1
0186*
0187*** SYSTEM AND BASE PAGE COMMUNICATIONS AREA ***
0188*
0189  01650                  EQU 1650B
0190*
0191*   I/O MODULE/DRIVER COMMUNICATION
0192*
0193  01660           EQT1   EQU .+8
0194  01661           EQT2   EQU .+9
0195  01662           EQT3   EQU .+10
0196  01663           EQT4   EQU .+11
0197  01664           EQT5   EQU .+12
0198  01665           EQT6   EQU .+13
0199  01666           EQT7   EQU .+14
0200  01667           EQT8   EQU .+15
0201  01670           EQT9   EQU .+16
0202  01671           EQT10  EQU .+17
0203  01672           EQT11  EQU .+18
0204  01771           EQT12  EQU .+81
0205  01772           EQT13  EQU .+82
0206  01773           EQT14  EQU .+83
0207  01774           EQT15  EQU .+84
0208*
0209*
      00105 000003
      00106 000001
      00107 000002
      00110 000004
      00111 000077
      00112 177400
      00113 000100
      00114 001100
      00115 004000
0210                         END
**  NO ERRORS*
```

PRIVILEGED INTERRUPT PROCESSING

When a special I/O interface card, HP 12610, is included in the system, RTE-C allows a class of privileged interrupts to be processed independently of regular RTE-C operation, with a minimal delay in responding to interrupts. The presence and location of the special I/O card is selected at system generation time by RTSGN. Its actual hardware location is stored in the word DUMMY in base page (or if not available, zero). See Section VI for the exact specification procedure.

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-C 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-C is running, and they can interrupt any RTE-C operation.

PRIVILEGED INTERRUPTS

The privileged interrupts can be recognized within 100 microseconds and processed in two ways:

a. Through a privileged driver which has in general the structure of a standard I/O driver plus a special privileged interrupt processor routine;

b. Through a special routine embedded in the system area.

Note that the routines which handle privileged interrupts must be completely independent of RTE-C.

If the first method is used, the calling program can make a standard I/O call to the privileged device. The calling program will be suspended for the time it takes to do the transfer, after which it will be rescheduled. To the calling program, there is no difference between a privileged type device I/O call and a non-privileged (standard) type device I/O call. If the privileged driver is assigned a time-out parameter, the parameter must be long enough to cover the period from I/O initiation to transfer completion.

If the second method is used, 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-C are not aware of these 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-C 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, sets the memory protect flag, MPTFL, equal to one, and checks the DUMMY flag. If the DUMMY 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 clears the control flip-flop on each DMA channel to defer DMA completion interrupts, and 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-C use this routine also. $IRT sets MPTFL = 0 to indicate that memory protect is "ON", and also 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, whether embedded directly within the system or within a privileged driver, must save and restore all registers, and restore memory protect to its original state (word MPTFL contains this status). This is because any interrupt automatically disables memory protect. The privileged interrupt routine must not use any features or requests of RTE-C, 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 the pre-defined locations in COMMON. The starting address of the 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.

SAMPLE PRIVILEGED DRIVER

The following discussion describes an example privileged driver, generalized to DVRXX, which is controlling a device operating in the privileged mode.

The device transfers one word of data each time it interrupts, and the data is stored into the buffer passed to the driver via the call parameters. Also passed to the driver is the number of data words to be input from the privileged device, this being the length of the data buffer.

The concepts behind such a driver are as follows:

It is called by a standard EXEC I/O call.

The calling program is placed into I/O suspension.

The device's trap cell is changed from "JSB CIC" to "JSB P.XX" where P.XX is the entry point to the privileged routine within the driver.

Therefore, each time the device interrupts, the RTIOC overhead is circumvented because the privileged routine P.XX is entered directly.

After each interrupt, if another data point is still required to satisfy the buffer length, the device is again encoded to subsequently interrupt, and the privileged routine is exited.

When the entire data buffer has been filled, the driver needs a way to communicate to the Executive that the transfer is complete. This is accomplished by allowing the driver to time-out. The time-out causes RTIOC to re-enter the driver at C.XX.

C.XX returns the transmission log, via the B-Register, and a successful completion indication, via the A-Register, to RTIOC.

RTIOC then reschedules the program which called the driver through its normal I/O completion machinery.

A standard RTE-C driver uses the EQT for all its temporary storage so that the same driver can be driving more than one device simultaneously. A privileged driver, however, cannot do this because it can never know the state of pointers to the EQT while it is running since it is running independently of the Executive. The privileged driver keeps its temporary storage internally, and therefore, can control only one device. For each device the driver will control, the driver must be reassembled with all names DVRXX and $JPXX (for this example) changed to another number. Then one driver per device must be loaded into the system at generation time.

INITIATION SECTION, I.XX—Refer to the partial listing of the sample privileged driver following this discussion. A standard I/O call to input from the device causes the calling program to be I/O suspended and the driver to be entered at I.XX. The request code is checked for validity.

Because this driver can control just one device (unlike standard drivers), there is no need to configure it more than once. Therefore, the first time the driver is entered, it is configured and the switch at "FIRST" is cleared so that on all subsequent entries, the configuration code is not executed.

The modification of the device trap cell is performed just once, after the configuring routine, and is not modified again on all later entries into the initiator. The trap cell is altered so that the device interrupts will be channeled to the P.XX subroutine instead of to RTIOC. The "JSB P.XX" instruction and its associated base page link are established via the small program "$JPXX" (see listing).

A counter, which is incremented in routine P.XX, is established for the number of readings to be taken; the buffer address for the storage of the data is saved, and the device is set up to initiate a reading and is encoded. The initiator then exits.

PRIVILEGED SECTION, P.XX—When the device interrupts, P.XX is entered as a result of the device's trap cell modification.

Because entry is made directly into P.XX the routine must do the housekeeping which RTIOC does when entered from an interrupt. Before P.XX can turn the interrupt system back on for higher priority interrupts, it must ensure that the DMA channels cannot interrupt, save the old memory protect status, and set its new status.

P.XX then loads and stores the data in the next unfilled buffer word. If there is yet another data point to be taken, P.XX sets up the device for the next reading, disables the interrupt system, encodes the device, restores memory protect status and its flag, turns the interrupt system back on, and exits. This basically resets the system to its state before P.XX was entered.

When the last reading is taken, P.XX disables the interrupt system, turns off the device, and sets up the driver for an immediate time-out. Before P.XX exits, it restores memory protect status and its flags, and turns the interrupt system back on.

COMPLETION SECTION, C.XX—The status of the device and the driver is now unchanged until the TBG interrupts. The TBG interrupt will cause a time-out (this is because bit 12 is set in EQT word 4), which will cause RTIOC to pass control to C.XX which returns a transmission log and a normal completion indication to RTIOC.

RTIOC then goes to its I/O completion section which reschedules the calling program and processes the device queue as if it were a standard (non-privileged) device.

```
JIMB2  T=00004 IS ON CR00105 USING 00012 BLKS R=0000

0001  ASMB,R,L,T,B
0002  *
0003  *
0004  *  DRIVER WITH PRIVILEGED INTERRUPT
0005  *
0006  *
0007         NAM DVRXX
0008         ENT I.XX,P.XX,C.XX
0009         EXT $JPXX
0010  *
0011  *
0012  *
0013  *  CALLING SEQUENCE:
0014  *      JSB EXEC       CALL EXEC
0015  *      DEF *+5        RETURN POINT
0016  *      DEF RCODE      REQUEST CODE
0017  *      DEF CONWD      CONTROL WORD
0018  *      DEF BUFFR      ADDRESS OF BUFFER
0019  *      DEF LENTH      LENGTH OF BUFFER
0020  *
0021  *
0022  *
0023  *
0024  *  CAUTION:  THIS DRIVER WILL NOT WORK WITH MORE THAN
0025  *      ONE SUBSYSTEM.  IF MORE THAN ONE SUBSYSTEM
0026  *      EXISTS IN A SYSTEM, BOTH DVRXX AND $JPXX MUST
0027  *      BE RE-ASSEMBLED WITH ALL THE NAMES CONTAINING
0028  *      'XX' CHANGED TO SOME OTHER NUMBER.  THEN ONE
0029  *      DRIVER PER SUBSYSTEM MUST BE PUT INTO THE SYSTEM
0030  *      AT GENERATION TIME.
0031  *
0032  *  INITIATION SECTION
0033  *
0034  I.XX   NOP
0035         STA SCODE      SAVE SELECT CODE
0036         LDA EQT6,I     REQUEST CODE TO A
0037         AND M77
0038         CPA =B1        READ REQUEST?
0039         JMP *+3         YES
0040  REJCT  CLA,INA         NO - ERROR
0041         JMP I.XX,I     REJECT RETURN
0042  FIRST  RSS            CONFIGURE FIRST
0043         JMP INIT        TIME ONLY
0044  *
0045         LDA SCODE
0046         IOR LIA        CONFIGURE
0047         STA IO0         IO INSTRUCTIONS
0048         .
0049         .
0050         .
0051         LDA $JPXX      SET TRAP CELL TO
0052         STA SCODE,I     JSB P.XX
0053         LDA EQT4,I     CLEAR EQT4 BIT12
0054         IOR BIT12       TO ALLOW NORMAL
0055         XOR BIT12       TIME OUT.
0056         STA EQT4,I
0057         LDA EQT15      SAVE EQT15
0058         STA EQ15        AND EQT4 ADDRESSES
```

```
0059          LDA EQT4       FOR LATER,
0060          STA EQ4
0061          CLA            SET SO AS NOT TO
0062          STA FIRST       CONFIGURE AGAIN
0063    *
0064    INIT  LDA EQT8,I     NUMBER OF CONVERSIONS TO A
0065          CMA,INA        NEGATE FOR
0066          STA CVCTR       CONVERSION COUNTER
0067          SSA,RSS        REJECT IF
0068          JMP REJCT       NUMBER <=0
0069          LDA EQT7,I     SAVE DATA BUFFER
0070          STA DAPTR       ADDRESS FOR P.XX
0071          JSB READ       START A READING
0072    IO1   STC IO,C       ENCODE DEVICE
0073          JMP I.XX,I     RETURN
0074    *
0075    READ  NOP            ROUTINE CONTAINING
0076          .               CONFIGURED IO
0077          .               INSTRUCTIONS TO
0078          .               SET UP THE DEVICE
0079          JMP READ,I      TO INITIATE ONE READING
0080    *
0081    *  PRIVILEGED INTERRUPT ROUTINE
0082    *
0083    P.XX  NOP
0084          CLF 0          TURN OFF INTERRUPTS
0085          CLC 6          TURN OFF
0086          CLC 7           DMA INTERRUPTS
0087          STA ASV        S
0088          STB BSV         A
0089          ERA,ALS          V
0090          SOC               E
0091          INA
0092          STA EOSV        REGISTERS
0093          LDA MPTFL      SAVE MEMORY
0094          STA MPFSV       PROTECT FLAG
0095          CLA,INA        TURN OFF MEMORY
0096          STA MPTFL       PROTECT FLAG
0097          STF 0          TURN ON INTERRUPTS
0098    *
0099          .              LOAD IN DATA
0100          .               FROM DEVICE
0101          .               VIA IO INSTRUCTIONS
0102    *
0103          STA DAPTR,I    STORE IN DATA BUFFER
0104          ISZ CVCTR      LAST CONVERSION
0105          RSS             NO
0106          JMP DONE        YES
0107          ISZ DAPTR      SET UP FOR
0108          JSB READ        NEXT CONVERSION
0109          CLF 0          TURN OFF INTERRUPTS
0110    IO4   STC IO,C       ENCODE DEVICE
0111    *
0112    EXIT  LDA MPFSV      WAS MEMORY
0113          SZA             PROTECT ON?
0114          JMP EXIT1       NO, FORGET DMA'S
0115          LDB INTBA      TURN
0116          LDA B,I         DMA'S
0117          SSA              BACK
0118          STC 6              ON
```

```
0119          INB                    IF
0120          LDA B,I                 THEY
0121          SSA                      WERE
0122          STC 7                     ON
0123  EXIT1   LDA EOSV        RESTORE
0124          CLO                E AND
0125          SLA,ELA             O
0126          STF 1                FLAGS
0127          LDB BSV           RESTORE B
0128          LDA MPFSV         RESTORE MEMORY PROTECT
0129          STA MPTFL          FLAG IN SYSTEM
0130          SZA               MEMORY PROTECT ON?
0131          JMP *+5            NO
0132          LDA ASV            YES, RESTORE A
0133          STF 0             TURN ON INTERRUPTS
0134          STC 5             SET MEMORY PROTECT
0135          JMP P.XX,I        RETURN
0136          LDA ASV           RESTORE A
0137          STF 0             TURN ON INTERRUPTS
0138          JMP P.XX,I        RETURN
0139  *
0140  DONE    CLF 0             TURN OFF INTERRUPTS
0141  IO7     CLC IO            TURN OFF DEVICE
0142          CCA               SET TIME OUT FOR
0143          STA EQ15,I         ONE TICK AND SET
0144          LDA EQ4,I          BIT12 IN EQT4 SO
0145          IOR BIT12          RTIOC WILL CALL
0146          STA EQ4,I          C.XX ON TIME OUT.
0147          JMP EXIT          GO TO EXIT
0148  *
0149  *
0150  *  COMPLETION SECTION
0151  *
0152  *
0153  C.XX    NOP
0154          CLA               SET UP FOR NORMAL RETURN
0155          LDB EQT8,I        TRANSMISSION LOG TO B
0156          JMP C.XX,I        RETURN
0157  *
0158  *
0159  *  CONSTANTS AND TEMPORARIES
0160  *
0161  *
0162  SCODE   OCT 0
0163  CVCTR   OCT 0
0164          .
0165          .
0166          .
0167  LIA     LIA 0
0168  M77     OCT 77
0169  DAPTR   DEF 0
0170  ASV     OCT 0
0171  BSV     OCT 0
0172  EQSV    OCT 0
0173  MPFSV   OCT 0
0174  EQ4     NOP
0175  EQ15    NOP
0176  BIT12   OCT 10000
0177  *
0178  *
```

```
0179   *  SYSTEM COMMUNICATION AREA
0180   *
0181   .      EQU 1650B
0182   INTBA  EQU .+4
0183   EQT4   EQU .+11
0184   EQT5   EQU .+13
0185   EQT7   EQU .+14
0186   EQT8   EQU .+15
0187   EQT15  EQU 1774B
0188   XA     EQU .+49
0189   XB     EQU .+50
0190   XEO    EQU .+51
0191   MPTFL  EQU .+80
0192   A      EQU 0
0193   B      EQU 1
0194          END
```

```
JIMB3  T=00004 IS ON CR00105 USING 00001 BLKS R=0000

0001   ASMB,R,L,B
0002           NAM $JPXX
0003           ENT $JPXX
0004           EXT P.XX
0005   $JPXX   JSB P.XX
0006           END
```

SECTION VI
RTE-C SYSTEM GENERATION

INTRODUCTION

This section is divided into two parts. Part 1 contains directions for planning or laying out an RTE-C System, and Part 2 uses the plans to configure an absolute RTE-C System on paper tape. General directions are provided for completing the RTE-C System Configuration Worksheet, and then transferring the plans into instructions to RTSGN. If the user is quite familiar with the RTSGN process (including the Loader section since many of the loader commands are used to relocate programs), a punched tape containing all the parameter inputs can be punched from the configuration worksheet, and then placed in the tape reader on the system teleprinter. Each time RTSGN requires an input from the operator the punched tape will provide the normal operator response. This method speeds up the system configuration considerably. If the user is not familiar with the RTSGN process, it is recommended that this section be read several times to completely familiarize him with all the steps required in a successful system configuration.

Appendix C contains an example of a completed worksheet and the teletype/system responses. This is an actual system configuration example intended to show the sequence of events to be expected in using the following planning process.

Part I
Instructions For Planning RTE-C

INPUT/OUTPUT PLANNING

The following instructions are keyed to the Configuration Worksheet. Fill in the blanks as you plan your system.

Input/output locations in all HP 2100 series computers have the same sequence of priority addresses: the highest priority address is the lowest numbered select code (I/O location). The octal select codes start at 10 in the computer mainframe and continue up to 67, with the HP 2150 series extender.

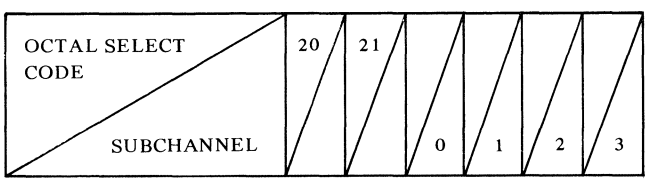
Interface cards are assigned to priority addresses according to the speed of interrupt response required by the I/O device. Interface cards for high-speed devices are assigned higher priority addresses than low-speed devices. For example, the time base generator interrupts every 10 milliseconds; therefore, its select code (I/O location) assignment must have a higher priority address than the teleprinter. Devices requiring privileged interrupt are always assigned to the highest priority addresses; while DMA devices are assigned the lowest.

STEP 1: I/O LOCATIONS

Considering the factors given in the preceding paragraphs and the instructions given below, select the I/O location for each I/O card, and fill in the top portion of the Input/Output Configuration Worksheet table with the I/O card name, and the appropriate select code (I/O slot).

NOTE

The top portion of the table is used for either the select code or the subchannel number. For example, if a cassette magnetic tape unit with four cassettes (four sub-channels) is connected to a controller in select codes 20 and 21, the top portion of the table would be completed thus.



This method of noting sub-channel numbers will facilitate assigning logical unit numbers later in the table.

The following detailed steps show how to assign select codes to devices starting at the highest priority address octal, select code 10 (octal). In addition to these steps, make certain that any peripheral devices or subsystems that use multiple select codes (I/O slots) have their I/O cards together and in the relative order required by that device or subsystem.

a. Assign all devices that require privileged interrupt in order of decreasing interrupt rate (i.e., the transfer speed in interrupts per second).
b. After the privileged devices, assign the privileged interrupt I/O card HP 12620.
c. Assign the TBG I/O card HP 12539.
d. Assign all devices that *do not* use DMA in order of decreasing speed.

NOTE

There will be occasions when a device uses DMA for data transfer and still generates an interrupt for end-of-record (EOR) processing. In these cases the hardware priority of the device should be treated as a non-DMA device, with the interrupt rate of the EOR condition determining its priority locations. Some consideration should be given to the priority of a data transfer vs. the priority of a record termination. Data transfers would normally be given priority over EOR interrupts or equivalent or even slightly slower interrupt rates.

e. Assign all devices that *do* use DMA in order of decreasing speed.
f. If an I/O extender is required and the extender does not have DMA capability, the order of steps "d" and "e" can be reversed so that all DMA devices are in the computer mainframe. If this step is necessary, maintain the same relative order of speed assignment among the DMA and non-DMA devices.

STEP 2: STANDARD LOGICAL UNIT ASSIGNMENTS

Make the standard logical unit number (LU) assignments (1 through 6) to I/O devices by placing an X at the intersection of the standard logical unit number, and the I/O card select code. Do not assign LU2 and LU3 to any device. This is to preserve upward compatibility with the Real-Time Executive Disc Based System.

STEP 3: ADDITIONAL LOGICAL UNIT ASSIGNMENTS

Starting with decimal 7, write in the logical unit numbers sequentially for each device or subchannel number as applicable. These numbers can be arbitrarily assigned to I/O devices, and do not have to be written in a left to right order on this table. Order is attained when RTSGN prints DRT TBL.

NOTE

If a device has two I/O cards, use only the highest priority I/O card for steps 2 and 3.

STEP 4: DRIVER IDENTIFICATION

Write in the driver identification number, obtained from the software box, for each device; e.g., multiple-device driver is DVR00. For devices or subsystems that have more than one I/O card, refer to the I/O card or subsystem documentation covering that device and driver. Place an "I" under the select code number of all I/O cards (i.e., every I/O card must have an entry in the interrupt tables). Place a dash under subchannel numbers.

STEP 5: DMA

Write in a large "D" for DMA required under each device that will use DMA. Note that some drivers are capable of dynamically assigning a DMA channel to themselves when required. Refer to individual driver documentation for more information on this capability.

STEP 6: EQT TABLE

Starting with decimal 1, write in the Equipment Table Entry (EQT) numbers sequentially for each device. DMA devices should be assigned EQT numbers in order of their DMA priority. A device that has subchannels is assigned the same EQT number for each subchannel.

STEP 7: BUFFERING

Write in a large "B" for devices that will use output buffering.

STEP 8: TIME-OUT

Write in a large "T" for devices that will use the time-out parameter. Values will be assigned later on the configuration worksheet.

RTE-C CONFIGURATION WORKSHEET

Once the system is planned, the actual configuration process takes place. This process is divided into four phases and must be executed in the order presented on the RTE-C Configuration Worksheet. The worksheet should be completed in advance from the information entered on the Input/Output Worksheet so your replies are complete and ready. The following general instructions will provide aid in completing the configuration worksheet. Actual operating instructions for RTSGN (sample configuration) are found in Part 2 of this section.

INITIALIZATION PHASE

RTSGN requires system specifications that broadly define the boundaries of the operating system. Specifications are required for the main input device, time base generator channel, privileged interrupt, and memory bounds. Some of the specifications dealing with memory bounds are automatically defaulted by the Loader portion of RTSGN while others must be supplied by the operator. It is possible for the operator to override the defaults using the Loader commands described in Section VII. For example, the Loader defaults the system starting address to octal 2000. If the user desires, this address can be changed during the Program Input Phase using the BOUNDS Command. Once the two extreme memory bounds have been established (upper bound by the boundary of the Basic Binary Loader, and lower bound by the last I/O location used), the generator will build the system within the given confines and establish all other boundaries. Examples of these boundaries are shown in Figure 6-1a which shows the system core map before generation, and 6-1b which shows the core map after generation.

Referring to Figure 6-1a, it can be seen that the system starts building upward from the boundary defined as the first word available for base page linkages. The area from FWA BP (established by the user) to 1647 is allocated to system linkages and user program linkages. The area from 1650 to 1777 is allocated to the system communication area. Starting at 2000 RTSGN relocates the system modules into absolute format. FWAM is initially set to 2000 by the Loader and is a pointer that is moved up by RTSGN as the modules are relocated. The system code takes up some of the area and what remains is allocated to the user. Note that the upper boundary is established by

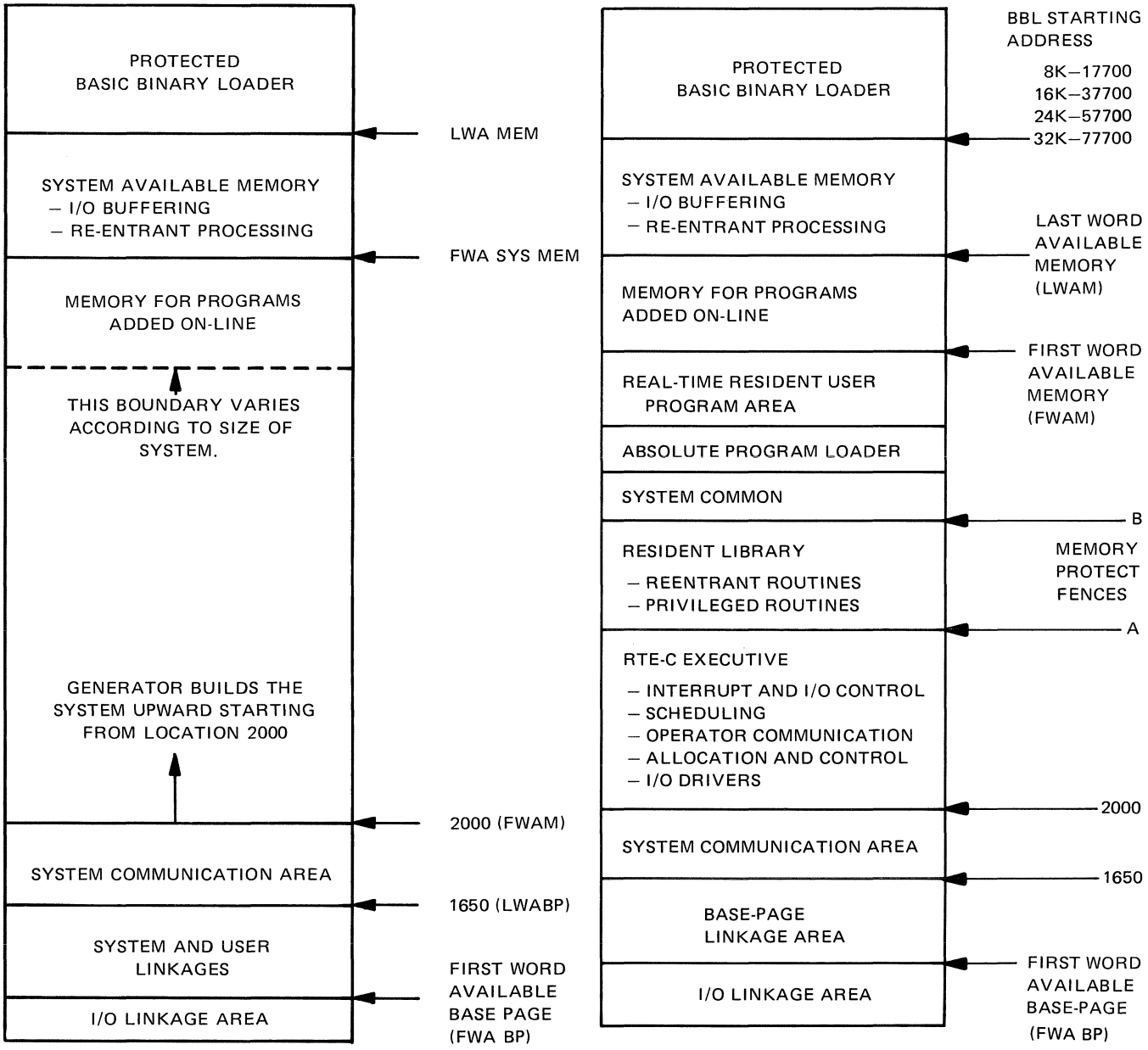


Figure 6-1. RTE-C Memory Maps

the user as FWA SYS MEM. After the system is generated, FWA SYS MEM becomes LWAM as shown in Figure 6-1b, and the area remaining between FWAM and LWAM is allocated as memory for programs added on-line in the future.

PRAM INPT?

Write in the logical unit number of the device (usually the system keyboard, LU1) that will be used for entering the parameters.

TBG CHNL?

Write in the I/O channel assignment (octal number) of the time base generator card.

PRIV. INT?

Write in the I/O channel assignment (octal number) of the privileged interrupt HP 12620 card.

FWA BP?

The first word available for base page linkages is established from the I/O Configuration Worksheet as the last used I/O location plus one (e.g., if the last I/O card in the priority string is located in I/O location 26, the FWA BP Linkage would be 27). This location will be used by RTSGN as the starting point in building the system. Determine the number from the worksheet and fill in the blank.

LWA MEM?

The last word of available memory core depends on computer core size.

8K – 17677	20K – 47677
12K – 27677	24K – 57677
16K – 37677	32K – 77677

FWA SYS MEM?

This is the first word of system memory available for re-entrant processing and I/O buffering and is shown as FWA SYS MEM in Figure 6-1a. However, after the system is configured, this boundary becomes the last word of available memory and is shown as LWAM in Figure 6-1b. This boundary is the upper boundary of the real-time area for user programs. Note that the APLDR will not allow a user's program to cross this boundary into the buffered I/O area. Remember that after this boundary is established, RTSGN builds the system up towards it. If improperly planned, the area remaining for adding on-line programs could be too small to be useful.

In selecting this boundary consider the following points:

- The larger the I/O buffer area, the faster programs can run that require I/O buffering.
- The larger the I/O buffer area means less area for user programs.
- Examine the user programs to determine the type of I/O buffering they may require. Programs that output large quantities of data to a line printer require a large I/O buffer, while programs that output relatively small amounts of data do not.

The generator has a default option that sets the FWA SYS MEM to the end of the last user program loaded. The default option allows the maximum amount of buffered I/O area to be declared by entering the default parameter 0 (zero). When entered, RTSGN completes loading the last user program before establishing the FWA SYS MEM boundary. The actual number can be obtained from the memory map under the High Main heading. Refer to the sample RTSGN in Appendix C.

NOTE

APLDR would not be able to add programs on-line if the default option is chosen.

PROGRAM INPUT PHASE

The program input phase relocates the system modules, drivers, and interrupt subroutines into an absolute format. The relocatable modules are loaded into the system in the following order.

I/O Control (RTIOC)
Scheduler (SCHED)
Exec Control (EXEC)
I/O Drivers (DVR00, etc.)
Interrupt Subroutines

TABLE GENERATION PHASE

There are three tables associated with this phase of RTSGN. Refer to the I/O Configuration Worksheet to obtain the required entries.

EQUIPMENT TABLE ENTRY (EQT TBL)

The first table to be completed is the Equipment Table. The information for each entry is located on the I/O Worksheet and is to be transferred to this table. The first entry is Equipment Table Entry number one and is usually

reserved for the device requiring the fastest system response. The blanks are filled in as follows:

octal select code, DVR number, D for DMA, B for buffered output, T for time-out, and time-out value.

Note that the parameters following the DVR number can be entered in any order, and missing parameters do not have to be accounted for.

The value for T must be a positive decimal number of up to four digits. This is then the number of time base generator interrupts (10 msec intervals) between the time I/O is initiated on the device and the time after which the device should have interrupted. (Note that for privileged drivers, T must be long enough to cover the period from I/O initiation to transfer completion.) If the device has not interrupted by this time, it is considered to have timed-out and is set-down, except in the case of the system teletype. For a device controlled by driver DVR00 (e.g., teleprinter), or DVR05 (DVR05 reserved for future system control device), T should not be less than 5000.

Example:

```
EQT 1 = ?
15    , DVR56  , D     ,      , T =

EQT 2 = ?
17    , DVR77  ,       ,      , T = 40

EQT 3 = ?
13    , DVR00  , B     ,      , T = 5000
```

This tells RTSGN that EQT entry number one relates to select code 15 and uses DVR56 with DMA required. EQT entry number two relates to select code 17 and uses DVR77 with time-out set to 400 milliseconds. EQT entry number three relates to DVR00 with output buffering and time-out set to 50 seconds.

DEVICE REFERENCE TABLE (DRT TBL)

The Device Reference Table, which contains the logical unit (LU) numbers, is cross referenced to the EQT entries here. Refer to the I/O worksheet to obtain the EQT entry number, LU number, and subchannel number if applicable. The first six LU numbers are reserved for system devices as follows:

LU1 – System teleprinter
LU2 – System mass storage (upward compatibility)
LU3 – Auxiliary mass storage (upward compatibility)
LU4 – Standard punch unit
LU5 – Standard input unit
LU6 – Standard list unit

Since LU2 and LU3 are reserved for upward compatibility with the Real-Time disc-based system, the operator must enter zeros to reserve these LUs.

INTERRUPT TABLE (INT TBL)

The interrupt links are tied to the select codes for input/output processing. Each I/O card (select code), in ascending order, is given an interrupt option. The format is:

select code, option, entry

Select code is entered from the I/O worksheet in ascending order.

Option directs RTE-C in handling the interrupt; there are four options:

select code, EQT, xx

The address of EQT entry numbered *xx* will be set into the interrupt entry associated with the *select code* location. The interrupt location will be filled with JSB $CIC,I.

select code, PRG, *name* causes program *name* to be scheduled upon interrupt.

select code, ENT, *entry* causes control to transfer to the entry point of a user-written system program upon interrupt.

select code, ABS, *xxxxx* places an absolute octal value (instruction code) in the interrupt location (may be NOP, CLC, etc.).

For devices or subsystems that have more than one I/O card, refer to the I/O card or subsystem documentation covering the device and driver. In all cases, each I/O card must have an interrupt entry. Note that interrupt location 4 (power fail) may be changed from its present HLT 4 to an ENT entry if a power-fail routine is to be included in the system. For example:

```
4, ENT, POWER
```

where POWER is an entry point in a user-supplied power-fail routine.

PROGRAM/PARAMETER INPUT PHASE

During the program portion of this phase, the user written programs and Absolute Program Loader (ADLDR) are loaded. The parameter portion establishes ID segments, a

start-up program, and number of words for common. The operator can also modify the name, priority, and execution interval of programs just loaded.

NUMBER OF ID SEGMENTS (# ID SEG?)

This entry establishes the number of ID segments required by the system. One ID segment is required for each user program loaded into the system during this generation, and for each program to be loaded on-line by the APLDR. If five ID segments are allocated, then only five programs in total can be loaded into the system. That includes programs loaded during generation (including the APLDR), and to be loaded later, on-line. If a program is deleted from the system by an OF, *name,* 8 operator command, the program's ID segment is returned to the system to use for another on-line load. Each ID segment occupies 28 words in the system area. Fill in the number of ID segments required. (Note: 0 (zero) ID segments is not allowed.)

START-UP PROGRAM (STRT-UP PROG?)

This entry creates a skeleton ID segment and puts it into the scheduled list. Whenever the RTE-C System is loaded into core and executed, the specified program is automatically scheduled for execution. The APLDR can be scheduled as the start-up program. Then when the system is loaded and executed, a list of all programs configured into the system will be listed. Write in the program's name and optional parameters; otherwise, if this feature is not desired, enter a 0 (zero).

RELOCATABLE RESIDENT LIBRARY (REL RES LIB)

Relocatable Resident Library (REL RES LIB) re-entrant or privileged subroutines (type 6) may be loaded at this point to be kept in a protected area of core. Use the following steps to determine the routines required and enter them on the worksheet.

STEP 1–Examine each program you intend on loading into the system and determine which library routines you will need. Use Appendix D as an aid.

STEP 2–Write the name of each routine into the blanks provided on the worksheet. The routines are listed in a first to last order in Appendix D, and should be entered in this order. Care taken in the order of entering the subroutine can avoid many rewinds of the library tape.

NUMBER OF WORDS IN COMMON (# WDS IN COMM?)

Allocate enough words to accommodate the largest COMMON statement in your programs. The decimal number may be zero (0) or positive.

RELOCATABLE USER PROGRAMS (REL USER PROGS)

User programs are entered at this point in the generation, as well as the APLDR if on-line program loading is desired. The programs are loaded in stages; the main program is loaded first followed by subprograms and then library routines. This stage is terminated by entering END. Following END, RTSGN allows you to change the program parameters if so desired. Be certain that all external referenced subroutines are known about. Once the END statement is entered it is not possible to go back and enter a forgotten subroutine. If you are not aware of required external subroutines, use the DISPLAY UNDEFS command.

ENTER PROGRAM PARAMETERS (ENTER PRAMS)

An ID segment is completed using the name and parameters of the program's NAM statement, or using the information supplied in this part. This entry provides the user the opportunity of changing the name, priority, or schedule parameters of the program just previously relocated by changing those parameters in the NAM statement. Default parameters are used if no parameters are specified in the NAM statement and none are supplied in response to ENTER PRAMS. The name of the ID segment will be the name of the first program relocated unless a name change is entered in response to the ENTER PRAMS request.

When the final user program is loaded, RTSGN asks if the Snapshot information module is desired. Snapshot contains ASCII information that provides the Relocating Loader with information on loader bounds, symbol table, and program linking. The Relocating Loader uses this information when relocating programs for the system. In other words, if you want to generate absolute programs off line using the loader, and then later load the programs into the system on-line, the Snapshot is required.

Part II
Generation Procedure

GENERAL INFORMATION

The set-up and operation of an RTE-C System involves two essential steps; the system must be configured into an absolute binary tape using the Real-Time System Generator, RTSGN (29101-60011), and the absolute binary tape must be loaded into the computer memory and executed.

This part describes the steps necessary to configure an RTE-C System, with the resulting absolute binary code punched on paper tape. All of the required information should be preplanned and located on the Input/Output and Configuration Worksheets filled out in Part 1. The answers used in this part are from an example RTE-C System that has been configured on worksheets per the instructions in Part 1. The completed worksheets and RTSGN listing are located in Appendix C.

The RTE-C Generator has incorporated within its code a portion of the Relocating Loader. Operation of the loader during generation is signified by the prompt character "-" (dash). Whenever this prompt is printed by the TTY, any loader command is available. Refer to Section 4 for loader information.

RTE-C GENERATOR

RTSGN operates on the same minimum hardware configuration as that required for an RTE-C System, and configures the system to fit a particular user's core memory size, I/O equipment, and programming needs.

To accomplish this, RTSGN requests certain initializing 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 punches them on paper tape. RTSGN creates I/O tables by identifying each I/O device and its associated driver routine, and establishes procedures for interrupt processing on each channel.

RTSGN is an absolute program on paper tape that is loaded into core by the Basic Binary Loader (BBL). Since RTSGN is independent of the RTE-C System which it generates, the I/O operations of RTSGN require SIO Drivers.

OPERATING PROCEDURES

The operation of RTSGN involves four phases.

a. Initialization Phase–RTSGN requires system specifications that broadly define the operating system. Responses are required to establish the select code (I/O channel) for the main input device, time base generator channel, privileged interrupt, and to establish memory bounds.
b. Program Input Phase–The operator loads the system modules, drivers, and interrupt subroutines. This phase does not allow any undefined externals to exist.
c. Table Generation Phase–The operator creates three tables; the Equipment Table, Device Reference Table, and Interrupt Table.
d. Program/Parameter Input Phase–The operator loads user written programs and, if required, the Absolute Program Loader (APLDR). 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.) RTSGN requests the number of ID segments to be reserved for subsequent program addition during the generation, and later for on-line loading.

NOTE

If, during any of the phases, the operator commits an non-recoverable error, the generation will be aborted. The procedure must then be repeated from the beginning.

To execute RTSGN and configure an RTE-C System, follow these steps:

a. Apply power to all equipment. Set the system teleprinter to LINE.
b. Load the RTSGN tape and configured SIO Drivers into core using the BBL.
c. Go to the starting address of the program, 100 (octal), and set the switch register as follows:

1. If the TTY punch is to be used, set bit 15 = "1". The following information halts will occur:

HLT 07 = turn punch on
HLT 70 = turn punch off

2. If the TTY punch is not used, set bit 15 = "0" (clear the switch register).

d. Push RUN. RTSGN begins the initialization phase.

INITIALIZATION PHASE

During initialization phase, RTSGN first establishes running operating parameters. After each question is printed, the operator responds with the required answer followed by carriage return/line feed (CR/LF). Operator responses are shaded and are only examples (see Appendix C); actual responses should be appropriate to the particular system being generated. If an error is made and discovered before LF is entered, type the RUBOUT key then CR/LF. Otherwise restart at step "c" above.

RTSGN requests the logical unit number of the input device. If the responses are to be entered through the system TTY enter LU1, if through the paper tape reader, LU5.

PRAM INPT?
1

RTSGN requests the octal select code of the time base generator.

TBG CHNL
13

RTSGN requests the octal select code of the HP 12620 Privileged Interrupt I/O card.

PRIV. INT?

Operator responds with the octal select code of the card if it is present (and all devices in higher priority slots become privileged), or a 0 (zero) if it is not.
0

RTSGN requests the first word of available core memory in base page.

FWA BP?

Operator responds with the first available octal select code number after the last I/O card.
26

RTSGN requests the last word of available core memory, which is the BBL address minus 1.

LWA MEM?
37677

RTSGN requests the first word available for system memory. This is the boundary between the user program area and buffered I/O area (refer to Figure 6-1).

FWA SYS MEM

Operator responds with 0 (zero) for default address, or an octal address somewhat below the last word of available memory. Refer to Part 1 for guidelines in determining this entry. The Appendix C response is:
36000

PROGRAM INPUT PHASE

During the program input phase, RTSGN accepts relocatable system modules and drivers from the library input unit (paper tape reader LU5). Relocatable programs should be loaded in the following order.

I/O Control (RTIOC)
Scheduler (SCHED)
Exec Control (EXEC)
I/O Drivers (DVR00, etc.)
Interrupt Subroutines

Before the modules are loaded, the map option of the relocating loader can be turned on using the loader MAP command. This will provide you with a complete core map of the system for future reference.

RTSGN prints:
REL SYS MODS
—

Operator can respond with the loader map command, or bypass the map option and proceed with loading. Our example uses the map option.

MAP GLOBALS, MODULES, LINKS

RTSGN responds by printing the core map headings across the page starting with Program Module and ending with High Base.

PROGRAM ..... HIGH
MODULE BASE
- -

Place the RTIOC module in the input device and type:

```
RELOCATE
```

Continue this process until all the modules have been loaded. To be certain that there are no undefined external references, type:

```
DISPLAY UNDEFS
```

If there are any undefined externals, they must be satisfied at this point or the generation will be aborted. Terminate the Program Loading Phase by typing:

```
END
```

RTSGN responds by printing the absolute starting address of the system and the links table if specified by MAP.

```
STARTING ADDRESS 00002
NO UNDEFS
LINK TABLE
   .     .
   .     .
```

```
TABLE GENERATION PHASE
```

If you make a mistake at any point in the table generation phase, the stage can be repeated by entering one of the following commands: RE to repeat the Equipment Table, RD to repeat the Device Reference Table and RI to repeat the Interrupt Table. If you go back more than one stage, then everything must be re-entered. For example, if you are in the DRT TBL and discover a mistake in the EQT TBL, type RE. The EQT TBL will be started over, followed by the DRT TBL. RTSGN requests the Equipment Table entries.

```
EQT TBL
```

Operator responds with a series of one line EQT entries which start with EQT # 1. The EQT entry relates the EQT number to an I/O channel and driver. Refer to the configuration worksheet and enter the Equipment Table entries. The Appendix C example is entered as follows:

```
EQT 1 = ?
15, DVR00

EQT 2 = ?
16, DVR01

EQT 3 = ?
17, DVR02, B, T = 1000

EQT 4 = ?
23, DVR62

EQT 5 = ?
END
```

RTSGN requests the logical unit assignments for the Device Reference Table.

DRT TBL

For each logical unit number, RTSGN prints:

n = EQT # ?

where:

n is a decimal integer starting with one.

The operator responds with an EQT entry number appropriate to the standard definition of *n*, and the subchannel number, if appropriate. Logical unit numbers 1 through 6 are predefined in the RTE-C System as:

1 – system teleprinter
2 – system mass storage (upward compatible)
3 – auxiliary mass storage (upward compatible)
4 – standard punch unit
5 – standard input unit
6 – standard list unit

Refer to the Configuration Worksheet and enter the Device Reference Table Entries. The Appendix C example is entered as follows:

```
1 = EQT # ?          (system teleprinter)
1
2 = EQT # ?          (system mass storage)
0
3 = EQT # ?          (auxiliary mass storage)
0
4 = EQT # ?          (standard punch unit)
3
5 = EQT # ?          (standard input unit)
2
6 = EQT # ?          (standard list unit)
1
7 = EQT # ?
4
8 = EQT # ?
END
```

RTSGN requests the Interrupt Table entries.

INT TBL

The operator responds with an entry for each I/O card, in ascending order (except I/O location 4). Refer to the Configuration Worksheet and enter the Interrupt Table Entries.

Note that the entry for location 4 (power-fail) may be entered out of order. This is the only location allowed out of order. The Appendix C example entries are as follows:

```
10, PRG, APLDR
15, EQT, 1
16, EQT, 2
17, EQT, 3
23, EQT, 4
END
```

PROGRAM/PARAMETER INPUT PHASE

This phase of RTSGN is where the user-written programs and utility library routines are configured into the system. If future absolute programs are to be loaded on-line, the APLDR will also be configured into the system (as a user program).

RTSGN requests the total number of ID segments required by the system for each program to be loaded.

#ID SEG?

Operator responds with a decimal number.

```
7
```

RTSGN requests the name of a program to be scheduled when the RTE-C System is loaded and run.

STRT-UP PROG?

Operator responds with the name of a program as it appears on the worksheet. If there is no start-up program enter 0 (zero).

```
APLDR
```

RTSGN next requests the Relocatable Resident Library.

REL RES LIB
–

Operator responds by placing the library tape into the input device and typing RELOCATE (*name*) where *name* is the first module listed on the worksheet that is to be loaded. After relocating the last module type:

```
END
```

RTSGN requests the number of words (in decimal) to be reserved for common storage.

#WDS IN COMM?

The Appendix C response is

```
100
```

The next state of this phase is the loading of user programs, the program specified for start up, and the APLDR, if required. RTSGN requests the loading of these programs:

REL USER PROGS
–

The operator must place the first program to be loaded into the tape reader and type:

```
RELOCATE
```

Since the Relocating Loader's map option is still enabled, the program's name and core bounds will be printed. Following this RTSGN will print the prompt character (dash) and wait for the next command. If there are external utility subroutines referenced by the program, they must be entered at this point with the RELOCATE or SEARCH command. If you do not know which subroutines are referenced, use the DISPLAY UNDEFS command, and they will be listed. When this stage of generation is complete (i.e., program has been loaded and there are no external subroutines referenced), the operator enters:

–

```
END
```

RTSGN now provides the operator with the opportunity of changing the just entered program's name, priority, or schedule parameters.

ENTER PRAMS

Operator responds with new parameters as entered on the worksheet, or a 0 (zero) if there are none.

RTSGN responds with a request for another user program.

REL USER PROGS

_

The operator must place the second program to be loaded into the tape reader and type:

RELOCATE

This procedure of loading programs and entering parameters continues until all programs are loaded. If a start-up program or any interrupt programs were declared in previous steps, now is the time for them to be loaded. To terminate this stage of generation type END following the command to relocate another program.

REL USER PROGS

_

END

At this time, RTSGN checks to see if the start-up program (if one was specified) was loaded. If not, a message is generated that informs you that the program is missing, and the program's name. For example:

STRT-UP PROG
APLDR
REL USER PROGS
_

The operator should place the program named APLDR in the tape reader and type:

RELOCATE

RTSGN will load the program and return the prompt character. If there are external subroutines referenced by the program they must be entered at this point with the RELOCATE or SEARCH command. If you do not know which subroutines are referenced, use the DISPLAY UNDEFS command and they will be listed. When this stage of generation is complete, type:

_

END

RTSGN now provides the operator with the opportunity of changing the just entered program's name, priority, or schedule parameters.

ENTER PRAMS

Operator responds with new parameters as entered on the worksheet, or a 0 (zero) if there are none.

0

If any programs were specified as interrupt programs, and they were not loaded during the program input phase, RTSGN reminds you of them and requests a decision on loading them. For example:

INT PRGS
TEST 1
IGNORE?

The operator must answer either YES or NO. If the answer is NO, RTSGN requests that the program be loaded. For example:

NO

REL USER PROGS

_

The operator must place the program to be loaded into the tape reader and type:

RELOCATE

RTSGN will load the program and return the prompt character. If there are external subroutines referenced by the program they must be entered at this point with the RELOCATE or SEARCH command. If you do not know which subroutines are referenced, use the DISPLAY UNDEFS command and they will be listed. When this stage of generation is complete type:

END

RTSGN now provides the operator with the opportunity of changing the just entered program's name, priority, or schedule parameters.

ENTER PRAMS

Operator responds with new parameters as entered on the worksheet, or a 0 (zero) if there are none.

0

RTSGN responds with a request for another user program.

REL USER PROGS

_

The operator must place the next program to be loaded (if there is one) into the tape reader and type RELOCATE again. If there are no more programs to be loaded, type:

END

RTSGN responds with:

SNAPSHOT?

Answer YES if the Snapshot is required or NO if it is not.

YES

Several inches of feed holes will separate the absolute system from the Snapshot. It is recommended that the Snapshot be removed from the end of the system tape and appropriately marked. This facilitates later loading of Snapshot in conjunction with the Loader. This completes the system generation.

RTSGN FINISHED

INITIATING RTE-C

When RTE-C has been configured onto paper tape, it is loaded into core using the Basic Binary Loader (BBL). Once loaded and initiated, by starting at address 2, it is ready to process user tasks.

OPERATING INSTRUCTIONS

a. Turn on all equipment.
b. Load the absolute system tape into core using the Basic Binary Loader. Starting address are:

17700 – 8K	047700 – 20K
27700 – 12K	057700 – 24K
37700 – 16K	077700 – 32K

c. Verify that the halt code is 102077, then go to the starting address of the program, octal 2.
d. Clear the switch register and push RUN.
e. When RTE-C has been loaded into core, the real-time clock is automatically started at time zero. The system prints the following message as a reminder to the operator that the clock can be set to current-day time.

SET TIME

f. The operator sets the clock to current-day time using the TM operator request as follows:

TM, *day, hour, min, sec.*

Any other request can also be typed.

ERROR HALTS

The following halts can occur during use of the Basic Binary Loader:

Halt Code	Cause	Recovery Action
102055	Address error	Right Tape? Loaded in tape reader backwards?
102011	Checksum error	Bad tape (torn or chad caught in a hole), or dirty tape reader.

GENERATION ERROR MESSAGES

The following messages may be printed on the teleprinter during execution of RTSGN.

BPG OV

Meaning: Base page memory has overflowed, or there is no area available for base page relocatable data or base page links.

Action: Irrecoverable error.

CKSM

Meaning: A checksum error was detected in the relocatable input.

Action: The operator is asked "BACKUP?" You must manually reposition the relocatable before replying. If the answer is YES, the Relocating Loader will re-read the last record. If the answer was NO, the Relocating Loader will abort.

CMND?

Meaning: The preceding statement contains syntactical errors.

Action: Enter the correct statement.

COM OV

Meaning: Common area has overflowed, or there is no area available for Common data.

Action: Irrecoverable error.

DU ENT

Meaning: Duplicate entry point. The entry point name is printed and the duplicate entry point is ignored.

Action: None Required.

ERR AD

Meaning: Invalid address.

Action: Enter valid address.

ERR CH

Meaning: Invalid channel number or incorrect sequence.

Action: Enter correct response.

ERR DR

Meaning: Invalid driver name.

Action: Enter correct response.

ERR DU

Meaning: Duplicate program name.

Action: Revise program by re-labeling the name (in response to the ENTER PRAMS message).

ERR EQ

Meaning: Invalid EQT Number.

Action: Enter valid number.

ERR IN

Meaning: Execution interval error.

Action: Enter valid execution time.

ERR LU

Meaning: Invalid logical unit number.

Action: Enter valid number.

ERR NA

Meaning: Parameter name error (no such program).

Action: Enter valid parameter statement.

ERR PA

Meaning: Invalid parameter.

Action: Enter valid parameter statement.

ERR PR

Meaning: Parameter priority error.

Action: Enter valid parameter statement.

ERR TB

Meaning: Symbol table overflow, number of ID segments exceeded.

Action: Irrecoverable error. Revise or delete programs.

The following errors may be generated by the Relocating Loader during each of the program input phases of RTSGN.

IL EXT

Meaning: A Data Block (DBL) record contains a reference to an external which has not been entered through a previous EXT record. Could also indicate a READ error, including skipping (see CKSM error).

Action: Irrecoverable error.

IL REC

Meaning: The last relocatable record contains an illegal record type code.

Action: Same as CKSM.

MEM OV

Meaning: Main memory has overflowed, or there is no area available for program relocatable data.

Action: Irrecoverable error.

REC SE

Meaning: A record sequence error has occurred.

Action: Irrecoverable error.

SYM OV

Meaning: The Relocating Loader's symbol table has overflowed. There is no room in available memory to allocate for a symbol definition.

Action: Irrecoverable error.

INPUT/OUTPUT CONFIGURATION WORKSHEET

RTSGN NUMBER _____ DATE _____ PREPARED BY _____

STD. LOGICAL UNIT NOS.	SC / SUB	10																														
	I/O INTERFACE CARD NAME																															
1 SYS. TTY																																
2 SYS. MASS STORAGE																																
3 AUX. MASS STORAGE																																
4 PUNCH OUTPUT																																
5 INPUT																																
6 LIST OUTPUT																																
7_{10} TO 63_{10}																																
DVR IDENT. (DVRxx)																																
DMA REQUIRED (D)																																
EQT ENTRY NO.																																
BUFFERED OUTPUT (B)																																
TIME-OUT (T)																																
OCTAL SELECT CODE / SUBCHANNEL		10																														

RTE-C – CONFIGURATION WORKSHEET

INITIALIZATION PHASE

RTSGN

PRAM INPT?

_____

TBG CHNL?

_____

PRIV. INT?

_____

FWA BP?

_____

LWA MEM?

_____

FWA SYS MEM?

_____

PROGRAM INPUT PHASE

REL SYS MODS

I/O CONTROL (RTIOC)

SCHEDULER (SCHED)

EXEC CONTROL (EXEC)

I/O DRIVERS AND INT. SUBROUTINES
(Write in Names)

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

END

STARTING ADDRESS 00002

NO UNDEFS

TABLE GENERATION PHASE

EQT TBL

EQT 1 = ?

_____, DVR ___, _____, _____, T = _____

EQT 2 = ?

_____, DVR ___, _____, _____, T = _____

EQT 3 = ?

_____, DVR ___, _____, _____, T = _____

EQT 4 = ?

_____, DVR ___, _____, _____, T = _____

EQT 5 = ?

_____, DVR ___, _____, _____, T = _____

EQT 6 = ?

_____, DVR ___, _____, _____, T = _____

EQT 7 = ?

_____, DVR ___, _____, _____, T = _____

EQT 8 = ?

_____, DVR ___, _____, _____, T = _____

EQT 9 = ?

_____, DVR ___, _____, _____, T = _____

EQT 10 = ?

_____, DVR ___, _____, _____, T = _____

END

DRT TBL

LU #

1 = EQT #? (SYSTEM TELEPRINTER)

_____, _____

2 = EQT #? (SYSTEM MASS STORAGE)

_____, _____

RTE-C – CONFIGURATION WORKSHEET (Continued)

TABLE GENERATION PHASE (Continued)

3 = EQT #? (AUXILIARY MASS STORAGE)

_____ , _____

4 = EQT #? (STANDARD PUNCH UNIT)

_____ , _____

5 = EQT #? (STANDARD INPUT UNIT)

_____ , _____

6 = EQT #? (STANDARD LIST UNIT)

_____ , _____

7 = EQT #?

_____ , _____

8 = EQT #?

_____ , _____

9 = EQT #?

_____ , _____

10 = EQT #?

_____ , _____

11 = EQT #?

_____ , _____

12 = EQT #?

_____ , _____

13 = EQT #I

_____ , _____

14 = EQT #?

_____ , _____

15 = EQT #?

_____ , _____

16 = EQT #?

_____ , _____

17 = EQT #?

_____ , _____

END

INT TBL

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

_____ , _____ , _____

END

RTE-C – CONFIGURATION WORKSHEET (Continued)

PROGRAM/PARAMETER INPUT PHASE

#ID SEG?

____

STRT-UP PROG?

________ (Write in Program's Name)

REL RES LIB
(Write in Names)

________ ________ ________ ________ ________

________ ________ ________ ________ ________

________ ________ ________ ________ ________

________ ________ ________ ________ ________

________ ________ ________ ________ ________

END

#WDS IN COMM?

____

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

______, ______, ______, ______, ______, ______, ______, ______

RTE-C – CONFIGURATION WORKSHEET (Continued)

PROGRAM/PARAMETER INPUT PHASE (Continued)

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____, _____, _____, _____, _____, _____, _____, _____

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____, _____, _____, _____, _____, _____, _____, _____

REL USER PROGS

________ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____, _____, _____, _____, _____, _____, _____, _____

REL USER PROGS

END

SNAPSHOT?

________ (Enter YES or NO)

RTSGN FINISHED

SECTION VII
RTE-C RELOCATING LOADER

GENERAL DESCRIPTION

The RTE-C Relocating Loader accepts relocatable object programs and converts the code into an absolute format suitable for incorporating into the RTE-C System. The absolute code is incorporated into the RTE-C System on-line with the Absolute Program Loader (APLDR).

RTE-C and the Loader do not operate concurrently in the same computer. The relationship of the Loader to other components of the system is shown in Figure 7-1. The source program shown in the figure must meet the format shown in Appendix F of this manual, or the format allowed by the RTE Disc Based System Relocating Loader (NAM record contains parameters). A summary of the functions performed by the Loader is shown in Table 7-1. A glossary of the mnemonics used throughout this section is shown in Table 7-2.

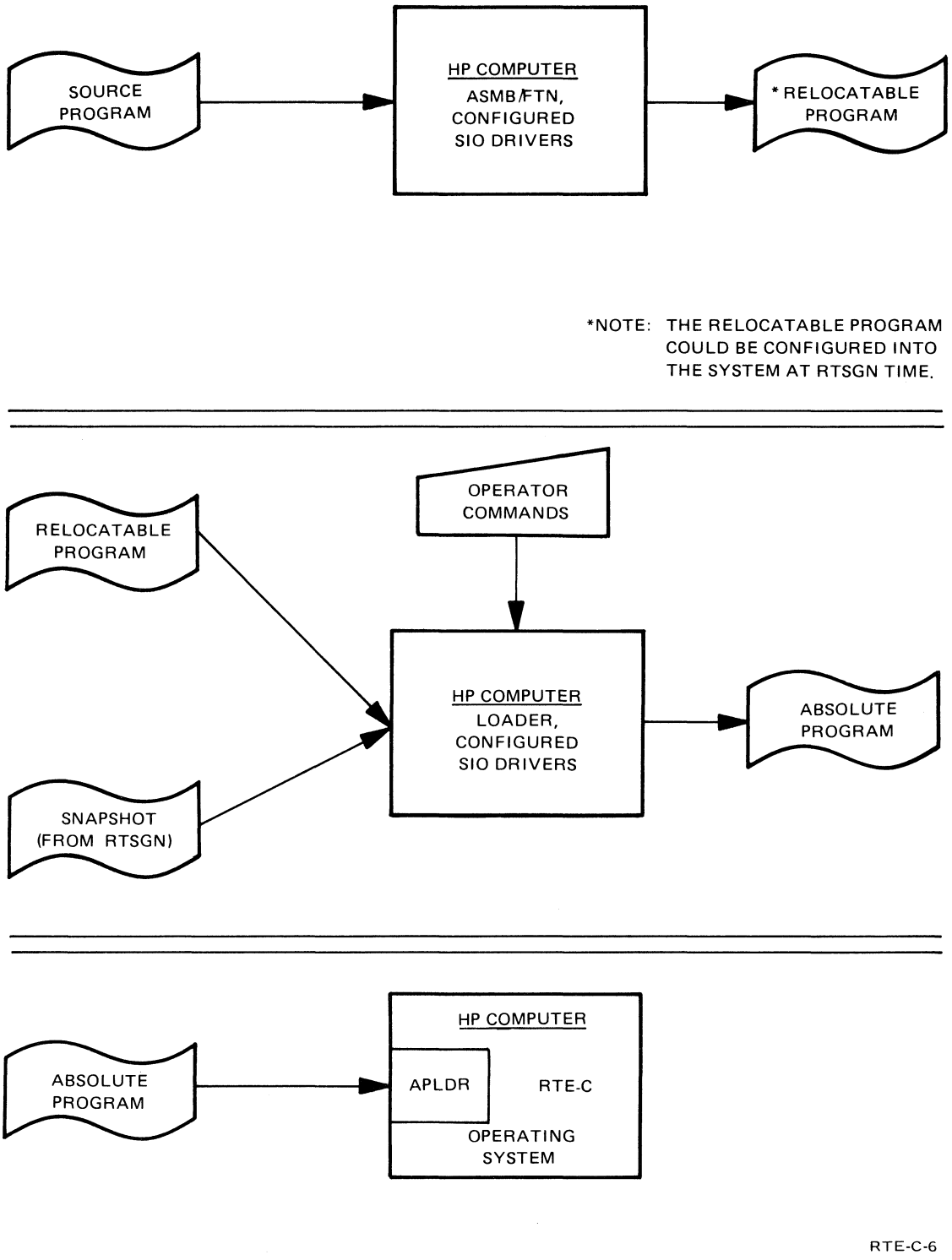


Figure 7-1. Loader To RTE-C System Relationship

Table 7-1. Summary of Loader Commands

Command	Function
*	All information following the asterisk, and before carriage-return/line-feed, is treated as a comment.
BOUNDS	Defines upper and lower memory bounds of absolute code generated by loader.
DISPLAY	Displays contents of symbol table, or names of undefined symbols, or values of defined symbols.
END	Specifies the end of the relocatable input phase.
LINKS START AT	This command transfers the base page linkage information, that was established during generation, to the Loader via the Snapshot tape. This command does not allocate a link, but tells of one already allocated.
MAP	Displays the system core map during the relocation process.
RELOCATE	Read and relocate a routine.
SEARCH	Provides selective relocation of modules that have entry points which satisfy undefined external references.
SET	This command transfers base page, current location counter and symbol table information, that was established during generation, to the loader via the Snapshot tape.
TRANSFER	Transfer input control to the tape reader.

Table 7-2. Glossary of Mnemonics Related to the Loader

Mnemonic	Meaning
FWAM	First Word of Available Memory for program relocatable data.
LWAM	Last Word of Available Memory for program relocatable data.
FWABP	First Word of Available Base Page memory for base page relocatable data.
LWABP	Last Word of Available Base Page memory for base page relocatable data.
FWAC	First Word of Available common memory for common data.
LWAC	Last Word of Available common memory for common data.
LOCC	Current Location Counter contains the address where a module may be relocated. LOCC is incremented after relocating each module to point to the next available core location. LOCC is initially set equal to FWAM.
BPLOCC	Base Page Location Counter contains address of the next available core location for relocating base page ORB data and base page links. BPLOCC is initially set equal to FWABP.
?XFER	Program Transfer Address is used by an operating system to determine the starting address of a program. If the END command is given and no transfer address was defined, the Relocating Loader will specifically request definition for ?XFER.

SNAPSHOT

The Loader operates hand-in-hand with the "Snapshot tape" produced by RTSGN during system generation. The Snapshot tape is located at the end of the RTE-C System absolute tape, and contains ASCII Loader commands required to directly build a copy of the RTSGN symbol table and set up new core bounds. This Snapshot provides the Loader with the following information:

- Boundary addresses of base page, user program memory area, and common. In addition, FWAM is set equal to LOCC, and FWABP is set equal to BPLOC. Note that if the Snapshot is not loaded, the Loader has default values for the boundaries. These default values are as follows:

FWAM = 2000	LWAM = 17677
FWABP = 100	LWABP = 1647
FWAC = 0	LWAC = 0

- The Snapshot contains some of the RTE-C executive entry points and all entry points of routines in the Resident Library area. Each entry point definition is in the form of a SET command to create a symbol table entry, and if appropriate, a LINKS START AT command to set up the link portion of the symbol table entry.

OPERATING INSTRUCTIONS

LOADING THE LOADER

The Loader is an absolute program that is loaded into a computer system with the Basic Binary Loader (BBL). To execute the Loader, proceed as follows:

a. Load the Loader into core with the BBL.
b. Load the configured SIO drivers into core with the BBL.
c. Go to the starting address of the Loader (100_8) and set the switch register to one of the following options.

1. Bit 15 = 1 means the teleprinter tape punch will be used to punch the absolute code. At appropriate times, the computer will halt to allow the operator to turn on or off the tape punch portion of the Model ASR33 Teleprinter (note that the ASR35 does not require switching when in KT Mode).
2. Bit 15 = 0 means the high speed tape punch will be used to punch the absolute code.

d. After setting the switch register option, press the PRESET and RUN buttons.
e. Place the RTE-C Snapshot tape (obtained from the system generation) in the tape reader and type TRANSFER. The Loader will load the Snapshot and ready itself for the relocating process.

RELOCATING A PROGRAM

The following is a generalized procedure for relocating a program.

a. Enable the core map options with the MAP Command.
b. Relocate the main module of the program with the RELOCATE Command.
c. Relocate any modules required by the main module. If a library of routines is used, the SEARCH command will relocate only those routines required. (i.e., entry points referenced by any previously relocated module.)
d. Use the DISPLAY UNDEFS Command to check for any undefined external references which may remain. Use either RELOCATE or SEARCH (as required) to relocate the necessary module unless it is desired to ignore the undefined reference.
e. Use the END Command to terminate the relocation process for this program.

MEMORY BOUNDS MANIPULATION

Note that the above operating procedure is very generalized and does not cover multiple program relocation or manipulating boundary addresses. Some of the criteria involved in manipulating boundary addresses are brought out in the following examples.

EXAMPLE 1 (MULTIPLE PROGRAM RELOCATION)

This example refers to the simplified memory map shown in Figure 7-2, and begins as follows:

a. The Snapshot generated with the system tells the Loader that the beginning of the user program area (FWAM) is address 25000. When Program A is relocated, "A's" beginning address will be set to 25000. Its ending address (+1 word) which may be used as the next FWAM, is obtained by the user from the core map generated during the relocation (see MAP Command). With Program A thus relocated, "A" can be loaded into the system on-line with the APLDR.
b. It is now desired to relocate Program B and load it into the system concurrently with program A. However, before going on there are some items that must first be taken into consideration.

- When the snapshot is initially loaded, it establishes a symbol table which contains a FWAM and FWABP for relocating a program. After the program is relocated, the Loader terminates. Therefore, when the Loader is restarted to relocate a second program, the symbol table and boundaries established by the previous Snapshot input are destroyed. This means that the snapshot must be reloaded.

- When the snapshot is reloaded the second time, it establishes the same symbol table and boundaries as before. Therefore, if Program B is loaded, it will start at the same locations Program A used and overlay Program A.

- To avoid this situation, the pertinent starting addresses must be reset using the BOUNDS Command after the Snapshot is loaded.

c. Obtain Program A's High Main address and High Base address from the core map and add one to each address. For example, Program A's High Main address could have been 27777 and its High Base address 00657. The BOUNDS Command could then be used as follows:

BOUNDS FWAM = 30000
BOUNDS FWABP = 00660

This resets FWAM and FWABP so that when Program B is relocated, "B's" beginning address is set to 30000, and its links begin at 00660.

d. In addition, location counters LOCC and BPLOCC must be set to the same value as FWAM and FWABP using the SET Command. As a shortcut to setting LOCC and BPLOCC (since in this case the information is redundant), set LOCC to zero. If LOCC is zero, LOCC is set to FWAM and BPLOCC to FWABP during execution of the first SEARCH or RELOCATE Command.
e. Program B can now be relocated with the RELOCATE Command.

EXAMPLE 2 (MANIPULATING COMMON)

This example refers to the simplified memory map shown in Figure 7-2, and begins as follows:

a. The Snapshot generated with the system tells the Loader that the boundaries of the system common area are FWAC = 10 and LWAC = 20. Programs A and B both use the system common area, but the area is not large enough for Program C to use. It is therefore desired to relocate Program C so that "C's" common area is internal to itself, and Program A and B can still use the system common area.
b. After relocating A and B according to the directions in Example 1, reload the Snapshot and update FWAM, FWABP, LOCC, and BPLOCC so Program C will not overlay Program A or B. Use the BOUNDS Command to alter the system common boundaries for Program C as follows:

BOUNDS FWAC = 0, LWAC = 0

c. This effectively tells the Loader that there is no system common area available.
d. Program C is now relocated with the RELOCATE Command. The default bounds for the common area are determined by the first module of Program C declaring a common area, and the common area is allocated before the code of that module is relocated. Therefore, the first module of Program C declaring common should declare the longest block used.

CONCLUSION FROM EXAMPLES

It is the Loader's responsibility to convert relocatable programs to absolute form, associate entry points with external references, and allocate base-page links as required. To this end, it is the user's responsibility to determine whether or not the current values of the core bounds are suitable for the module to be relocated. This can be accomplished by using the DISPLAY Command to show the values of the six bounds keywords (see Table 7-2) and if required, changing the bounds using the BOUNDS Command. Taking this precaution will insure that the absolute program will not overlap or overlay code of other programs which may be present in the RTE-C system when this program is loaded by APLDR. Actually APLDR performs checks to prevent an accidental overlay of programs (see Section 2, Table 204).

LOADER COMMANDS

The operator controls the Loader by requests entered through the system input device (refer to Table 7-1). Once the Loader is loaded and running properly, a hyphen (-) prompter is returned on the initial keyboard input device. This directs the operator that the Loader is ready to accept a command. Each command from a device other than a teletype (TTY) (high-speed tape reader) must be preceded by a hyphen (-). The general command syntax is shown in Table 7-3. The Backus Naur Form syntax of each command is presented in Table 7-5 at the end of this section.

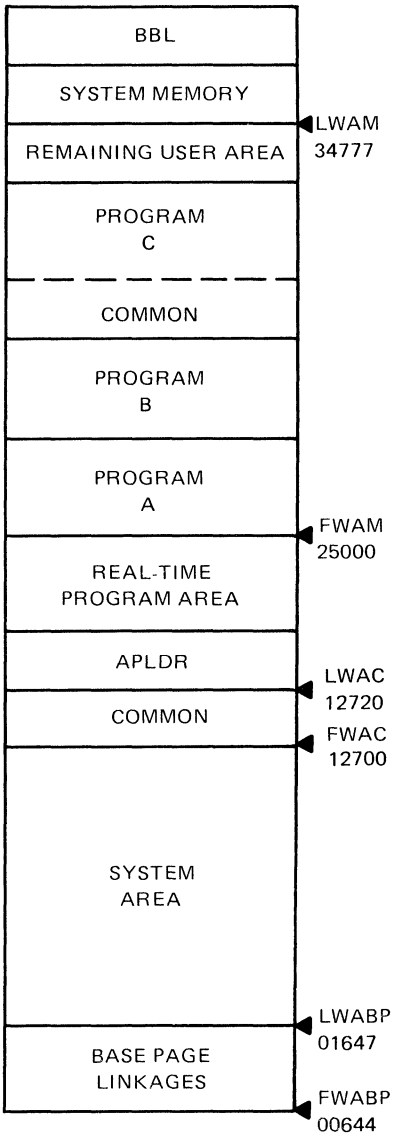


Figure 7-2. Simplified RTE-C Core Map

Table 7-3. Conventions in Operator Command Syntax

Item	Meaning
UPPER CASE ITALICS	These words are literals and must be specified as shown.
lower case italics	These are symbolic representations indicating what type of information is to be supplied. When used in text, the italics distinguishes them from other textual words.
[item]	Items with brackets are optional.
, (comma)	Some of the commands can have more than one literal (in any order) specified. When more than one literal is used the comma (,) separator is required.
item 1 *item 2* *item 3*	This indicates that there is a choice of entries for the parameter, but one parameter must be specified.

BOUNDS

Purpose:

Defines upper and lower memory bounds of absolute code generated by loader.

Format:

BOUNDS FWAM=n , LWAM=n , FWABP=n , LWABP=n , FWAC=n , LWAC=n

Where:

FWAM = First Word Available Memory
LWAM = Last Word Available Memory
FWABP = First Word Available Base Page
LWABP = Last Word Available Base Page
FWAC = First Word Available Common
LWAC = Last Word Available Common
n = octal address

COMMENTS

The mnemonics shown in the format statement can be entered in any order or combination. The only requirements are that at least one boundary be entered, and that multiple entries be separated by a comma.

If any part of a module needs to be relocated outside the current bounds, an overflow error message will be given. However, the module may reference symbols outside the available bounds (e.g. entry points to executive routines).

Default values for the program memory area (FWAM/LWAM) and base page area (FWABP/LWABP) are defined within the Loader and may be examined using the DISPLAY Command. The default values for the common area (FWAC/LWAC) are determined by the first module declaring it during relocation, and the common area is allocated before the code of that module is relocated. Therefore, the first module declaring common should declare the largest block used. The user can, however, create a larger area than required or force common to be at the high end of memory. This can be done by assigning appropriate addresses to FWAC and LWAC prior to the program relocation. The size of common is equal to LWAC-FWAC +1.

DISPLAY

Purpose:

Displays contents of symbol table, or names of undefined symbols, or values of defined symbols.

Format:

DISPLAY *TABLE*
UNDEFS
name
mnemonic

Where:

TABLE	specifies the symbol table contents.
UNDEFS	specifies undefined symbols.
names	specifies the name of a module or library routine.
mnemonic	specifies one of the three pairs of bounds, current/base page location counter, or program transfer address. Refer to Table 7-2.

COMMENTS

All information is printed on the keyboard input device. If the contents of the symbol table are printed with the DISPLAY TABLE commands, the format printed as follows:

name xxxxx

Note that if name is not in the table, the message will be:

name UNDEFINED

If *name* is in the table, but not associated with an entry point, the message will be:

name = 77777

END

Purpose:

To terminate the relocatable input phase.

Format:

END

COMMENTS

The END Command unconditionally terminates the relocating phase. If there are any undefined references, they will be set to zero. When the command is entered, punching of the base page links and ID segment information of the program is completed.

If it is desired to relocate another program and not overlay the one just previously relocated, the BOUNDS Command must be used to redefine the First Word of Available Memory (FWAM) and First Word of Available Base Page (FWABP).

LINKS START AT

Purpose:

To transfer base linkage information that was established during RTSGN into the loader.

Format:

LINKS START AT *link address, link value*

Where:

link address = octal address
link value = octal value

COMMENTS

This command is used by the operator to provide the loader with minimal information. The command does not create a link, but passes information for symbol table entries. The Snapshot tape is usually listed by the operator with the TTY off-line. The operator can then pick only the relevant information desired to satisfy his programming requirements (i.e., the entire Snapshot tape does not have to be input).

MAP

Purpose:

To display the system core map during the relocation process.

Format:

MAP *MODULES, GLOBALS, LINKS, OFF*

Where:

MODULES = modules' name.
GLOBALS = modules' entry point.
LINKS =base page linkage information.
OFF =turn MAP Command off.

COMMENTS

The MAP Command, if issued prior to the relocation process, provides a core map with headings and bounds as shown in Figure 7-3. The mnemonics shown in the Format statement can be entered in any order or combination. The only requirement is that at least one option be entered, and that multiple entries be separated by a comma.

```
REL SYS MODS
-
MAP LINKS,GLOBALS,MODULES
 PROGRAM  ENTRY    LOW     HIGH    LOW     HIGH
  MODULE  POINT    MAIN    MAIN    BASE    BASE
  ----------------------------------------------
-
RELOCATE
  EXEC            02000   03043   00026   00042
            $ABRT 02757
            $ALC  02437
            $ERMG 02702
            $LIBR 02205
            $LIBX 02314
            $RQST 02002
            $RTN  02534
            EXEC  02000
.
.
-
END

STARTING ADDRESS
NO UNDEFS
 LINKS TABLE
$CHTO  00077
$DEVT  00100
 .
 .
```

Legend:

Program Module	=	The name of the module being relocated.
Entry Point	=	All entry points referenced in the module.
Low Main	=	Starting address of the module. It is also used to show the value associated with each entry point.
High Main	=	Last memory location used by the module.
Low Base	=	First base-page link used.
High Base	=	Last base-page link used.

NOTE

If the High Base figure is one lower than the Low Base figure, no base-page area was used by that module.

Figure 7-3. Example MAP Command Printout

RELOCATE

Purpose:

To unconditionally relocate a module.

Format:

RELOCATE [*(name)*]

Where:

(name)	is the name of the module to be relocated. Note that name must be enclosed with parenthesis.

COMMENTS

The RELOCATE Command directs the Loader to unconditionally read and relocate a module from the standard input device. If the module name is given, only that module will be relocated, all others are skipped. (i.e., all preceding modules are skipped and the tape stops after relocating the named module.)

After the module has been loaded, the DISPLAY UNDEFS Command is usually used to obtain the undefined external references. The undefined references are satisfied with the SEARCH Command and the process is terminated with the END Command.

SEARCH

Purpose:

To provide selective relocation of modules that have entry points which satisfy undefined external references.

Format:

SEARCH [*(name)*]

Where:

(name)	is the name of the module or routine to be relocated. Note that *(name)* must be enclosed with parenthesis.

COMMENTS

The SEARCH *(name)* Command is usually used to relocate a certain library routine that was referenced in a module previously relocated. The routine is relocated only if it has an entry point name which matches a name previously declared as an external, and not yet associated with an entry point. The loader stops the input tape at the end of the named module (whether or not the module is relocated).

If the *name* option is not used, then the loader automatically searches the library tape and loads all routines which satisfy the external references.

SET

Purpose:

To transfer base page and current location counter information that was established during RTSGN into the loader.

Format:

```
        LOCC
SET     BPLOCC  TO octal add
        name
```

Where:

LOCC	=	Current location counter.
BPLOCC	=	Base page location counter.
name	=	The name of a module or library routine.
octal add	=	an octal number (either an address or value).

COMMENTS

This command is used by the operator to provide the loader with minimal information. The command may not be used after a relocation to satisfy undefined externals. That is, the command doesn't transfer information to any links that have been allocated, it only transfers existing information to the symbol table. The Snapshot tape is usually listed by the operator with the TTY off-line. The operator can then pick only the relevant information desired to satisfy his programming requirements (i.e., the entire Snapshot tape does not have to be input).

TRANSFER

Purpose:

To transfer input control to the tape reader.

Format:

TRANSFER

COMMENTS

All commands punched on the tape must be preceded by the hyphen (-). Comments are preceded by the asterisk (*).

SPECIAL ABSOLUTE RECORDS

The absolute code produced by the Loader must contain a series of special absolute records following the last relocated record (just before the end-of-file). These records are used by APLDR to build the program's ID segment. These records contain the program name, its type, priority, time values, and core bounds. Some of this information must be defaulted by the Loader if they were not supplied in the NAM record. Table 7-4 shows the order of these records, the absolute address, and the content of the record. Default values, if any, are also shown.

Table 7-4. Special Absolute Records

Record	Absolute Address	Data	Default
1	2	Word 1 of name	First NAM
	3	Word 2 of name	encountered
2	2	Word 3 of name and program type	Type = 1
	3	Program Priority	99
3	2	Time Res. Exec. Mult.	0
	3	Spare Word	
4	2	Hours	0
	3	Minutes	0
5	2	Seconds	0
	3	Tens of Milliseconds	0
6	2	Low Main	
	3	High Main	
7	2	Low Base Page	
	3	High Base Page	
8	2	First Address of Common	0
	3	Size of Common	0
9	2	JMP 3,I	
	3	Transfer Address	

The absolute code produced by the Loader can also be loaded into the computer with the Basic Binary Loader. The last record is used by the BBL to set up the starting address at octal location 2.

ERROR MESSAGES

The following messages may be printed on the teleprinter during operation of the Loader.

BPG OV

Meaning: Base page memory has overflowed, or there is no area available for base page relocatable data or base page links.

Action: Relocating Loader aborts.

CKSM

Meaning: A checksum error was detected in the relocatable input.

Action: The operator is asked "BACKUP?"; if the operator's answer is YES, the operator must manually reposition the relocatable before replying. If the answer was NO, the Relocating Loader will abort.

CMND?

Meaning: The preceding statement contains syntactical errors. A transfer is made to the teleprinter to allow corrections.

Action: Type the correct statement and, if necessary, TR to return to the command file.

COM OV

Meaning: COMMON area has overflowed.

Action: Relocating Loader aborts.

DU ENT

Meaning: Duplicate entry point.

Action: The entry point name is printed, then newer entry is ignored.

IL BND

Meaning: A bound less than 0 was entered in a BOUNDS command.

Action: Handled the same as "CMND?".

IL BPL

Meaning: Illegal base page length. This occurs when the base page length specified in the NAM record is less than zero. Relocatable records produced by the SIO ALGOL compiler fit into this category and thus may not be relocated by the RTE-C Loader.

Action: Relocating Loader aborts.

IL EXT

Meaning: A Data Block (DBL) record contains a reference to an external which has not been entered through a previous EXT record. Could also indicate a READ error, including skipping (see CKSM).

Action: The Relocating Loader Aborts.

IL REC

Meaning: The last relocatable record contains an illegal record type code. (This error also occurs when the Loader reads off the end of the tape while skipping leader.)

Action The operator is asked if backing up to the start of the record is desired. The response is the same as in CKSM error above.

MEM OV

Meaning: Main memory has overflowed, or there is no area available for program relocatable data.

Action: Relocating Loader aborts.

REC SE

Meaning: A record sequence error has occured.

Action: The operator is asked "BACKUP?"; if the answer is "YES" the operator will reposition the tape back one record before replying. The loader will "fake" an end record and continue by reading the next record. If the answer is "NO" the Relocating Loader will abort.

REL AB

Meaning: Whenever relocation is aborted, this message is printed at the teletype.

Action: The Loader has been cleared and restarted. Input to the loader as if it has just been started.

SYM OV

Meaning: The Relocating Loader's symbol table has overflowed. There is no room in available memory to allocate for a symbol definition.

Action: Relocating Loader aborts.

?XFER?

Meaning: No transfer address was found when the END command was given.

Action: The operator is asked "?XFER?". The reply should be an octal number to be used as the transfer address.

Table 7-5. Backus Naur Form of Loader Commands

Legend

::=	"is defined as . . . "
|	"or"
< >	enclose an element
" "	enclose a literal
[ ]	enclose an optional element

Syntactic Elements

```
<digit> ::=        0|1|2|3|4|5|6|7
<letter> ::=       A|B . . . . . .Y|Z
<spchar> ::=         $|%|&|?|!|.
<letsp> ::=        <letter> | <spchar>
<keyword> ::=      LOCC | BPLOCC | ?XFER | FWAM | LWAM | FWABP | LWABP | FWAC | LWAC
<integer> ::=      <digit> | <integer>  <digit>
<identifier> ::=   <letsp>|<identifier><letsp>|<identifier>  <digit>
```

BOUNDS

```
BOUNDS  <bounds assignment list>
<bounds assignment list>  ::=  <bounds keyword>  =  <integer>  [ , <bounds assignment list>  ]

<bounds keyword>  ::=  FWAM | LWAM | FWABP | LWABP | FWAC | LWAC
```

DISPLAY

```
DISPLAY  <display option>
<display option>  ::=  TABLE|UNDEFS|<identifier>|< keyword>
```

END

```
END
```

LINKS START AT

```
LINKS START AT  <link address>,<link value>
<link address>  ::=  <integer>
<link value>  ::=  <integer>
```

MAP

```
MAP  <map option list>
<map option list>  ::=  <map option keyword>  [, <map option list >]
<map option keyword>  ::=  MODULES|GLOBALS|OFF|LINKS
```

(Continue on next page)

Table 7-5. Backus Naur Form of Loader Commands (Continued)

RELOCATE

RELOCATE [(<module name>)]
<module name> ::= <identifier>

SEARCH

SEARCH [(<module name>)]

SET

SET <set dest> TO <integer>
<set dest> ::= LOCC | BPLOCC | <identifier>

TRANSFER

TRANSFER | TR

APPENDIX A
TABLES

This Appendix contains several useful Tables and Diagrams.

EQUIPMENT TABLE

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

Table A-1. 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		S	T		Unit #						Channel #				
5	AV		EQUIP TYPE CODE							STATUS							
6	CONWD (Current I/O Request Word)																
7	Request Buffer Address																
8	Request Buffer Length																
9	Temporary Storage for Optional Parameter																
10	Temporary Storage for Optional Parameter																
11	Temporary Storage for Driver																
12	Temporary Storage for Driver																
13	Temporary Storage for Driver																
14	Device Time-Out Reset Value																
15	Device Time-Out Clock																

DEVICE REFERENCE TABLE

The Device Reference Table (DRT) has an entry for each physical unit defined in the Equipment Table (EQT). The DRT consists of one-word entries corresponding to the range of user-specified logical units 1 to n, where n is less than or equal to 63_{10}.

The contents of the word in the DRT corresponding to a logical unit number is defined as follows:

Bits 0 - 5 = EQT number
Bits 11 - 13 = Subchannel number

A value of zero for the EQT number bits means that logical unit number is unassigned.

Table A-2. Device Reference Table

Logical Unit Number	Assignment
1	System teleprinter
2	System mass storage (upward compatibility)
3	Auxiliary mass storage (upward compatibility)
4	Standard punch unit
5	Standard input unit
6	Standard list unit
7 . . 63	Defined by user

PROGRAM ID SEGMENT

Each user program has a 28 word ID segment located in the system area. The address of each ID segment is located in the Keyword Table (see Appendix B). The ID segment contains static and dynamic information defining the properties of a program. The static information is set during RTSGN time or when a program is loaded on-line, and the dynamic information is maintained by the Executive.

The number of ID segments contained in a system is set during RTSGN time, and is directly related to the number of programs that can be in core at any given time. If all the ID segments are in use, no more programs can be added on-line.

The information contained in the ID segment is shown in Table A-3.

Table A-3. Program ID Segment

Word	Contents																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	LIST LINKAGE																
2	5-WORD																
3	BLOCK																
4	FOR																
5	PASSING																
6	PARAMETERS																
7	PRIORITY																
8	ENTRY POINT																
9	POINT OF SUSPENSION																
10	A-REGISTER																
11	B-REGISTER																
12	E-AND-O-REGISTERS																
13	NAME (1)										NAME (2)						
14	NAME (3)										NAME (4)						
15	NAME (5)										–					TYPE	
16			A	B	–	C	–	D	–							STAT	
17	TIME LIST LINKAGE																
18	RES.		E	MULTIPLE													
19	TENS OF MILLISECONDS																
20	SECONDS																
21	MINUTES																
22	HOURS																
23	LOW MAIN BOUND																
24	HIGH MAIN BOUND																
25	LOW BASE PAGE BOUND																
26	HIGH BASE PAGE BOUND																
27																	
28																	

Where:

A = 1 Waiting for schedule program completion.

B = 1 Program to be aborted after current suspension.

C = 1 Program to be operator suspended after current suspension.

D = 1 Program to be made dormant after current suspension.

E = 1 Program is in time list.

NAM RECORD USED IN RTE/DOS

Table A-4 shows the NAM record recognized by the RTE-C System.

Table A-4. NAM Record

Word	Contents																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	WORD COUNT = 21_8															0	
2	0010 ◄——► 0 Ident																
3	CHECKSUM																
4	NAME (1)										NAME (2)						
5	NAME (3)										NAME (4)						
6	NAME (5)										blank						
7	AC	PROGRAM LENGTH															
8	0 ◄——► 0																
9	'COMMON' LENGTH																
10	TYPE																
11	PRIORITY																
12	RESOLUTION CODE																
13	EXECUTION MULTIPLE																
14	HOURS																
15	MINUTES																
16	SECONDS																
17	TENS OF MILLISECONDS																

A/C = 0 assembler generated
1 compiler generated

APPENDIX B
SYSTEM COMMUNICATION AREA

A block of storage in base page, starting at octal 1650 contains the system communication area and is used by RTE-C 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., addresses less than octal 2000) 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
1650	EQTA	FWA of equipment table
1651	EQT No.	No. of EQT entries
1652	DRT	FWA of device reference table
1653	LUMAX	No. of logical units (in DRT)
1654	INTBA	FWA of interrupt table
1655	INTLG	No. of interrupt table entries
1656	–	Reserved for system
1657	KEYWD	FWA of keyword table
1660	EQT1	Addresses of First 11 words In Current EQT Entry
1661	EQT2	
1662	EQT3	
1663	EQT4	
1664	EQT5	
1665	EQT6	
1666	EQT7	
1667	EQT8	
1670	EQT9	
1671	EQT10	
1672	EQT11	
1673	CHAN	Current DMA channel no.
1674	TBG	I/O address of time-base card
1675	SYSTY	EQT entry address of system TTY
1676	RQCNT	No. of request parameters -1
1677	RQRTN	Return point address
1700	RQP1	Addresses of Request Parameters (Set for maximum of 8 Parameters)
1701	RQP2	
1702	RQP3	
1703	RQP4	
1704	RQP5	
1705	RQP6	
1706	TQP7	
1707	RQP8	
1710	DORMT	Address of 'dormat' list
1711	SKEDD	'Schedule' list
1714	SUSP3	'Available memory' list
1715	–	Reserved for system
1716	SUSP5	'Operator suspend' list
1717	XEQT	ID segment addr. of current program.
1720	XLINK	'Linkage'
1721	XTEMP	'Temporary' (5 words)
1726	XPRIO	'Priority' word
1727	XPENT	'Primary entry point'
1730	XSUSP	'Point of suspension'
1731	XA	'A-Register' at suspension
1732	XB	'B-Register'
1733	XEO	'E and overflow'
1734	OPATN	Operator/keyboard attention flag
1735	OPFLG	Operator communication flag
1736	–	Reserved for system
1737	DUMMY	I/O address of dummy int. card
1740	–	Reserved for system
1741	–	Reserved for system
1742	BPA1	FWA R/T BP link area
1743	BPA2	LWA R/T BP link area
1744	–	Reserved for system
1745	LBORG	FWA of resident library area
1746	RTORG	FWA of real-time area
1747	RTCOM	Length of real-time common area
1750	RTDRA	FWA of real-time area
1751	AVMEM	FWA of system available memory
1752	BKORG	LWA of system available memory
1753	–	
1754	–	
1755	–	
1756	–	
1757	–	
1760	–	Reserved for system
1761	–	
1762	–	
1763	–	
1764	–	
1765	–	
1766	–	
1767	–	
1770	MPTFL	Memory protect on/off flag (0/1)
1771	EQT12	Addresses of last 4 words in current EQT entry
1772	EQT13	
1773	EQT14	
1774	EQT15	
1775	FENCE	MEM protect fence address
1777	BKLWA	LWA of memory

APPENDIX C
EXAMPLE RTE-C GENERATION

```
RTSGN

PRAM INPT?
1

TBG CHNL?
13

PRIV. INT?
0

FWA BP?
26

LWA MEM?
37677

FWA SYS MEM
36000

REL SYS MODS
-
MAP MODULES,GLOBALS
 PROGRAM  ENTRY    LOW     HIGH    LOW     HIGH
  MODULE  POINT    MAIN    MAIN    BASE    BASE
  ----------------------------------------------
-


RELOCATE
  EXEC             02000   03022   00026   00042
            $ABRT  02737
            $ALC   02417
            $ERMG  02662
            $LIBR  02205
            $LIBX  02314
            $RQST  02002
            $RTN   02514
            EXEC   02000
  SCHED            03023   06360   00043   00221
            $CVT3  05772
            $LIST  03700
            $MPT1  06206
            $MPT2  06214
            $MPT4  06222
            $MPT5  06233
            $MPT6  06302
            $MPT7  06323
            $TREM  03407
            $XEQ   03440
            $CLCK  03023
            $MESS  04173
```

```
              $STRT   06056
              $TADD   03365
              $TYPE   06137
  RTIOC               06361    11041    00222    00311
              $IORQ   06575
              $IRT    06527
              $SYMG   10126
              $CHTO   10700
              $DEVT   07761
              $EQST   10603
              $IOCL   10312
              $IODN   10045
              $IOUP   10105
              $LUPR   10430
              $XSIO   07302
              $CIC    06361
              $CVEQ   10223
              $XCIC   06375
  DVR00               11042    12205    00312    00347
              C.00    11413
              C.01    11413
              C.02    11413
              I.00    11042
              I.01    11042
              I.02    11042
-
RELOCATE (DVR62)
  DVR62               12206    13060    00350    00347
              I.62    12206
              C.62    12407
-
END

STARTING ADDRESS 00002
NO UNDEFS

EQT TBL


EQT  1 =?
15,DVR00

EQT  2 =?
16,DVR01

EQT  3 =?
17,DVR02,B,T=1000

EQT  4 =?
23,DVR62

EQT  5 =?
END
```

```
DRT TBL
LU#

 1 = EQT #?
1

 2 = EQT #?
Ø

 3 = EQT #?
Ø

 4 = EQT #?
3

 5 = EQT #?
2

 6 = EQT #?
1

 7 = EQT #?
4

 8 < EQT #?
END

INT TBL

1Ø,PRG,APLDR

15,EQT,1

16,EQT,2

17,EQT,3

23,EQT,4

END


#ID SEG?
7
STRT-UP PROG?

APLDR
REL RES LIB
```

```
-
RELOCATE (GETID)
  GETID              13516    13657    00351    00352
             PUSH    13527
             POP     13542
             IDPTR   13571
             IDBUF   13574
             INTID   13516
-

RELOCATE (LIBT)
  LIBT               13660    13713    00353    00352
             LIBT    13660
-
END

STARTING ADDRESS 00002
NO UNDEFS
# WDS IN COMM?
100
REL USER PROGS
-
RELOCATE
  DUMMY              14060    14064    00353    00353
-
RELOCATE
  R2313              14065    14625    00354    00354
             R2313   14117
             A2313   14343
             B2313   14456
             S2313   14067
             ?QUE?   14502
             ?LU?    14420
-
RELOCATE
  P2313              14626    14726    00355    00354
             P2313   14636
-
RELOCATE
  R2930              14727    15241    00355    00354
             R2930   14744
-

RELOCATE
  D2313              15242    15527    00355    00354
             D2313   15253
             E2313   15457
-
DISPLAY UNDEFS
 UNDEFS
.ENTR
```

```
-
SEARCH
  .ENTR              15530    15617    00355    00354
             .ENTR   15537
             .ENTP   15530
-
END

STARTING ADDRESS 14060
NO UNDEFS
ENTER PRAMS




DUMMY,1
REL USER PROGS
-
END

STRT-UP PROG

APLDR

REL USER PROGS
-
RELOCATE
  APLDR              15620    17317    00355    00442
             APLDR   15620
-
RELOCATE (GETAD)
  GETAD              17320    17335    00443    00442
             GETAD   17320
             ADRES   17334
-
SEARCH
  RMPAR              17336    17360    00443    00442
             RMPAR   17336
-
END

STARTING ADDRESS 15620
NO UNDEFS
REL USER PROGS
-
MAP LINKS
-
END
SNAPSHOT?
Y
RTSGN FINISHED
```

RTE-C – CONFIGURATION WORKSHEET

INITIALIZATION PHASE

RTSGN

PRAM INPT?

1

TBG CHNL?

13

PRIV. INT?

0

FWA BP?

26

LWA MEM?

37677

FWA SYS MEM?

36000

PROGRAM INPUT PHASE

REL SYS MODS

I/O CONTROL (RTIOC)

SCHEDULER (SCHED)

EXEC CONTROL (EXEC)

I/O DRIVERS AND INT. SUBROUTINES
(Write in Names)

DVR00 _____

DVR62 _____

_____ _____

_____ _____

_____ _____

END

STARTING ADDRESS 00002

NO UNDEFS

TABLE GENERATION PHASE

EQT TBL

EQT 1 = ?

15 , DVR00, _____ , _____ , T= _____

EQT 2 = ?

16 , DVR01 , _____ , _____ , T= _____

EQT 3 = ?

17 , DVR02, B , _____ , T=1000

EQT 4 = ?

23 , DVR62 _____ , _____ , T= _____

EQT 5 = ?

END , DVR ___ , _____ , _____ , T= _____

EQT 6 = ?

_____ , DVR ___ , _____ , _____ , T= _____

EQT 7 = ?

_____ , DVR ___ , _____ , _____ , T= _____

EQT 8 = ?

_____ , DVR ___ , _____ , _____ , T= _____

EQT 9 = ?

_____ , DVR ___ , _____ , _____ , T= _____

EQT 10 = ?

_____ , DVR ___ , _____ , _____ , T= _____

END

DRT TBL

LU #

1 = EQT # ? (SYSTEM TELEPRINTER)

1 , _____

2 = EQT # ? (SYSTEM MASS STORAGE)

0 , _____

RTE-C – CONFIGURATION WORKSHEET (Continued)

TABLE GENERATION PHASE (Continued)

3 = EQT #? (AUXILIARY MASS STORAGE)
0 , _____

4 = EQT #? (STANDARD PUNCH UNIT)
3 , _____

5 = EQT #? (STANDARD INPUT UNIT)
2 , _____

6 = EQT #? (STANDARD LIST UNIT)
1 , _____

7 = EQT #?
4 , _____

8 = EQT #?
END , _____

9 = EQT #?
_____ , _____

10 = EQT #?
_____ , _____

11 = EQT #?
_____ , _____

12 = EQT #?
_____ , _____

13 = EQT #1
_____ , _____

14 = EQT #?
_____ , _____

15 = EQT #?
_____ , _____

16 = EQT #?
_____ , _____

17 = EQT #?
_____ , _____

END

INT TBL

10 , PRG , APLDR

15 , EQT , 1

16 , EQT , 2

17 , EQT , 3

23 , EQT , 4

END , _____ , _____

END

RTE-C – CONFIGURATION WORKSHEET (Continued)

PROGRAM/PARAMETER INPUT PHASE

#ID SEG?

7

STRT-UP PROG?

APLDR (Write in Program's Name)

REL RES LIB
(Write in Names)

GETID _____ _____ _____ _____

LIBT _____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

END

#WDS IN COMM?

100

REL USER PROGS

DUMMY (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

DUMMY, 1 , _____ , _____ , _____ , _____ , _____ , _____

REL USER PROGS

R2313 (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

REL USER PROGS

P2313 (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

REL USER PROGS

R2930 (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

REL USER PROGS

D2313 (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

REL USER PROGS

.ENTR (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

REL USER PROGS

APLDR (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____ , _____ , _____ , _____ , _____ , _____ , _____ , _____

RTE-C – CONFIGURATION WORKSHEET (Continued)

PROGRAM/PARAMETER INPUT PHASE (Continued)

REL USER PROGS

GETAD (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____, _____, _____, _____, _____, _____, _____, _____

REL USER PROGS

RMPAR (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____, _____, _____, _____, _____, _____, _____, _____

REL USER PROGS

_____ (Write in Program's Name)

END

ENTER PRAMS

NAME [, PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

_____, _____, _____, _____, _____, _____, _____, _____

REL USER PROGS

END

SNAPSHOT?

YES (Enter YES or NO)

RTSGN FINISHED

INPUT/OUTPUT CONFIGURATION WORKSHEET

RTSGN NUMBER _____ DATE _____ PREPARED BY _____

STD. LOGICAL UNIT NOS.	SC / SUB	10	11	12	13	14	15	16	17	20	21	22	23
	I/O INTERFACE CARD NAME	JUMPER CARD	JUMPER CARD	JUMPER CARD	TIME BASE GEN.	JUMPER CARD	TELEPRINTER I/O	TAPE READER I/O	TAPE PUNCH I/O	JUMPER CARD	JUMPER CARD	JUMPER CARD	2313 SUBSYSTEM I/O
1	SYS. TTY						X						
2	SYS. MASS STORAGE												
3	AUX. MASS STORAGE												
4	PUNCH OUTPUT								X				
5	INPUT							X					
6	LIST OUTPUT												
7_{10} TO 63_{10}													
DVR IDENT. (DVRxx)													
DMA REQUIRED (D)													
EQT ENTRY NO.													
BUFFERED OUTPUT (B)									B				
TIME-OUT (T)									T				
OCTAL SELECT CODE / SUBCHANNEL		10	11	12	13	14	15	16	17	20	21	22	23

APPENDIX D
RTE-C RELOCATABLE LIBRARY

There are three libraries or collections of relocatable subroutines that can be used by RTE-C: the RTE/DOS Relocatable Library, EAU version (F2E. *n*), and the RTE/DOS FORTRAN IV Library (F4D. *n*), and the RTE/DOS Floating Point Library (F2F. *n*). Refer to Figure D-1. The F4D. *n* Library contains a formatter, extended precision, and other routines that reference routines from the F2E. *n* Library. If extended precision is not required but the formatter is, a separate RTE/DOS Basic FORTRAN Formatter (FF. *n*) is available.

NOTE

The *n* referenced in the above paragraph represents the revision code letter.

The F2E. *n* Library also contains mathematical and utility routines such as SIN, COS, 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.

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-C through entry point $LIBR to save the data. At the end of execution, the re-entrant routine calls RTE-C through entry point $LIBX to restore the temporary data, if any.

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

NOTE

A library (type 6) program can only call another library program or system (type 0) program.

Format of Re-Entrant Routine

```
            EXT   $LIBR, $LIBX

ENTRY       NOP               Entry point of routine
            JSB   $LIBR       Call RTE-C to save
                              temporary data
            DEF   TDB         Address of
                              temporary data
            .                 Program instructions
            .
            .
EXIT        JSB   $LIBX       Call RTE-C to
                              restore data
            DEF   TDB
            DEC   n           n is for routines with
                              two return points in
                              the calling
                              program; 0 specifies
                              the error-print
                              return and 1 the
                              normal return. For
                              routines with only
                              one return point, n =
                              0.
TDB         NOP               System Control Word
            DEC   n+3         Total length of
                              current block
            NOP               Return address to
                              calling program
T1          }
.           }                 Temporary data
.           }                 (n words)
Tn          }
```

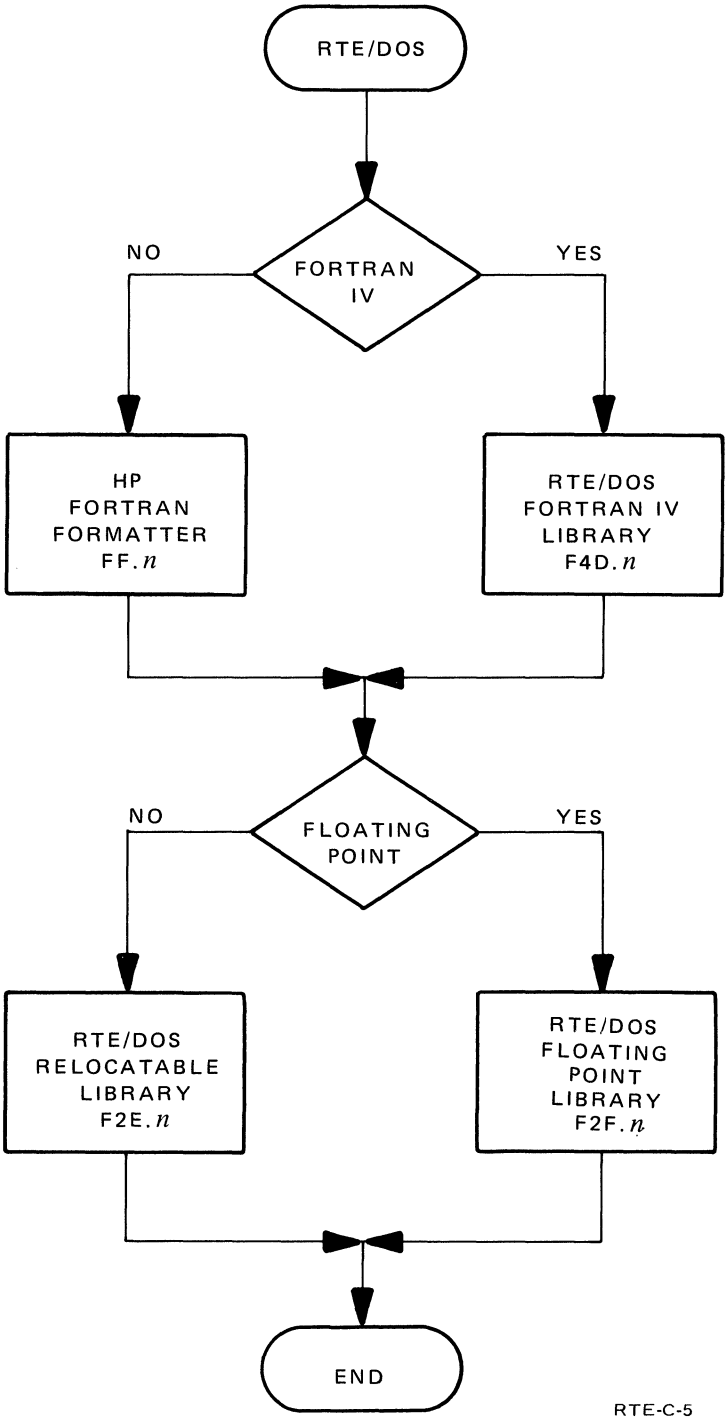


Figure D-1. RTE-C Library Configuration Diagram

PRIVILEGED SUBROUTINE STRUCTURE

Privileged subroutines execute with the interrupt system turned off. This feature allows many programs to use a single privileged subroutine without incurring re-entrant overhead. As a result, privileged subroutines 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

```
        EXT   $LIBR, $LIBX

ENTRY   NOP             Entry point to the
                        routine

        JSB   $LIBR     Call RTE-C to
                        disable the interrupt
                        system

        NOP             Denotes privileged
                        format
        .
        .
        .

EXIT    JSB   $LIBX     Call RTE-C 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 examples of utility routines.

When RTSGN creates the RTE-C System, all library subroutines not included in the resident library are loaded immediately following each user program during the Relocatable User Program (REL USER PROG) portion of the generation.

LIBRARY CORE REQUIREMENTS

A user-written HP FORTRAN program requiring formatted I/O, can use either the RTE/DOS Formatter or the FORTRAN IV Formatter. The following information on the formatters is provided as an aid in designing the memory allocation for the system generation process. The figures are approximate and have been rounded up to accommodate future changes.

The resident library core requirements will vary depending on which formatter is used. This is because each formatter calls different subroutines are shown below.

a. Using RTE/DOS Formatter and RTE/DOS Library, the following privileged routines are used:

Routine		Approximate Core Requirement	
IFIX	.ZRLB		
FLOAT	.OPSY	360_8	245_{10}
.FLUN	.EAU.		
.PACK			

b. Using the FORTRAN IV Library and RTE/DOS Library, the following privileged or re-entrant routines are used:

Routine		Approximate Core Requirement	
FRMTR	IFIX		
DBLE	FLOAT		
SNGL	.FLUN		
.XPAK	.PACK	3625_8	1940_{10}
.XCOM	.ZRLB		
.XFER	.OPSY		
.ENTR	.EAU.		

c. Using RTE/DOS Formatter and the Floating Point Library, the following privileged routines are used:

Routine		Approximate Core Requirement	
FLIB	.ZRLB		
.FLUN	.OPSY	405_8	261_{10}
.PACK	.EAU.		

d. Using the FORTRAN IV Library and the Floating Point Library, the following privileged routines are used.

Routine		Approximate Core Requirement	
FRMTR	FLIB		
DBLE	.ENTR		
SNGL	.FLUN	3565_8	1909_{10}
.XPAK	.PACK		
.XCOM	.ZRLB		
.XFER	.DPSY		
	.EAU.		

If a user FORTRAN program does not require formatted I/O, and uses the RTE/DOS Library as a minimum, the minimum routines that may be required are math routines such as FADSB, .EAU., FLOAT, etc.

The core requirements in the user program area will also vary depending on which formatter is used. Remember that a utility routine is appended to each program that requires it.

a. Using RTE/DOS Formatter and RTE/DOS Library, FRMTR requires 2545_8 or 1380_{10} words.

b. Using the FORTRAN IV Library and RTE/DOS Library, FMTIO) requires 1235_8 or 670_{10} words.

NOTE

The utility routine CLRIO is also used but only accounts for three words.

A conclusion that can be drawn from the above information indicates the following:

a. Using RTE/DOS Formatter

Resident Library	=	240	(Re-entrant/ Privileged)
FRMTR	=	+1380	(Utility)
1 program	=	1620_{10}	
		+1380	(Utility)
2 programs	=	3000_{10}	
		+1380	(Utility)
3 programs	=	4380_{10}	

b. Using FORTRAN IV Formatter

Resident Library	=	1940	(Re-entrant/ Privileged)
FMTIO	=	+670	(Utility)
1 program	=	2610_{10}	
		+670	(Utility)
2 programs	=	3280_{10}	
		+670	(Utility)
3 programs	=	3950	

Comparing "a" to "b" show that one program using formatted I/O uses less core when the RTE/DOS Formatter is used. However, three or more programs in the system that formatted I/O should use the FORTRAN IV Formatter to realize a savings in core.

SUBROUTINE STRUCTURE

Table D-1 has been included to aid the user in determining which routines are privileged, re-entrant or utility, and the order in which the routines are presented on the library tapes.

Table D-1. Order of FORTRAN Library Routines

Table D-1 applies to the following library revision codes:

RTE/DOS Relocatable Library HP 24151D

RTE/DOS FORTRAN IV Library HP 24152C

RTE/DOS FORTRAN Formatter HP 24153B

RTE/DOS Floating Point Library HP 24248B

RTE/DOS Relocatable Library HP 24151D

Routine	Privileged	Re-Entrant	Utility	Routine	Privileged	Re-Entrant	Utility
F2E.C							
CLRIO			x	.ITOI	x		
%ANH			x	ISIGN	x		
%XP			x	IABS	x		
%IN			x	CHEBY		x	
%OS			x	MANT	x		
%AN			x	PTAPE			x
%BS			x	MAGTP			x
%LOG			x	.ENTR	x		
%QRT			x	IFIX	x		
%IGN			x	FLOAT	x		
%LOAT			x	.FLUN	x		
%FIX			x	.PACK	x		
%TAN			x	..DLC	x		
%ABS			x	.GOTO			x
%SIGN			x	IAND			x
%AND			x	IOR			x
%OR			x	OVF			x
%OT			x	ISSW			x
%SSW			x	.MAP.			x
GETAD			x	RMPAR			x
TANH		x		CODE			x
.RTOR		x		ENTIE			x
TAN		x		.SWCH			x
EXP		x		.PRAM			x
SICOS		x		INDEX			x
SQRT		x		%WRIS			x
SIGN	x			PAUSE			x
ALOG		x		ERR0			x
.IENT	x			BINRY			x
ABS	x			SREAD			x
ATAN		x		%WRIT			x
PWR2	x			.ZRLB	x		
FDV	x			.OPSY	x		
FMP	x			.TAPE			x
..FCM	x			DEBUG			x
FADSB	x			DBKPT			x
.RTOI				.EAU.	x		

Table D-1. Order of FORTRAN Library Routines (Continued)

RTE/DOS FORTRAN IV Library HP 24152C							
Routine	Privileged	Re-Entrant	Utility	Routine	Privileged	Re-Entrant	Utility
F4D.C							
FMTIO			x	MOD	x		
FRMTR		x		AINT	x		
%INT			x	INT			x
%NT			x	IDINT	x		
%LOGT			x	DDINT		x	
$SQRT			x	MXMNI		x	
$LOGT			x	MXMNR		x	
$LOG			x	MXMND		x	
$EXP			x	DSIGN		x	
#COS			x	DIM	x		
#SIN			x	IDIM	x		
#LOG			x	.CFER			x
#EXP			x	..CCM			x
.RTOD			x	..MAP			x
.DTOR			x	.IDBL			
.DTOD		x		.ICPX			
DEXP		x		.DCPX			
ALOGT			x	.DINT			
DLOGT			x	.CINT			
DLOG		x		.CDBL			
DATN2		x		REAL	x		
DATAN		x		AIMAG	x		
DCOS		x		CMPLX	x		
DSIN		x		CONJG	x		
XPOLY		x		DBLE		x	
ENTIX		x		SNGL	x		
DSQRT		x		XADSB		x	
CLOG		x		XMPY		x	
ATAN2		x		XDIV		x	
CSQRT		x		CADD		x	
CABS		x		CSUB		x	
CEXP		x		CMPY		x	
CSNCS		x		CDIV		x	
DMOD		x		..DCM		x	
.DTOI		x		.XPAK		x	
.CTOI		x		.XCOM	x		
DABS		x		.XFER	x		
AMOD	x			.PCAD	x		

	RTE/DOS FORTRAN Formatter HP 24153B		
Routine	Privileged	Re-Entrant	Utility
FRMTR			x

Table D-1. Order of FORTRAN Library Routines (Continued)

RTE/DOS Floating Point HP 24248B							
Routine	Privileged	Re-Entrant	Utility	Routine	Privileged	Re-entrant	Utility
F2F.B							
CLRIO			x	.ITOI	x		
%ANH			x	ISIGN	x		
%XP			x	IABS	x		
%IN			x	CHEBY		x	
%OS			x	MANT	x		
%AN			x	PTAPE			x
%BS			x	MAGTP			x
%LOG			x	.ENTR	x		
%QRT			x	.FLUN	x		
%IGN			x	.PACK	x		
%LOAT			x	..DLC	x		
%FIX			x	.GOTO			x
%TAN			x	IAND			x
%ABS			x	IOR			x
%SIGN			x	OVF			x
%AND			x	ISSW			x
%OR			x	.MAP			x
%OT			x	RMPAR			x
%SSW			x	CODE			x
GETAD			x	ENTIE			x
TANH		x		.SWCH			x
.RTOR		x		.PRAM			x
TAN		x		INDEX			x
EXP		x		%WRIS			x
SICOS		x		PAUSE			x
SQRT		x		ERRO			x
SIGN	x			BINRY			x
ALOG		x		SREAD			x
.IENT	x			%WRIT			x
ABS	x			.ZRLB	x		
ATAN		x		.OPSY	x		
PWR2	x			.TAPE			x
FLIB	x			DEBUG			x
..FCM	x			DBKPT			x
.RTOI		x		.EAU.	x		

APPENDIX E
SUMMARY OF OPERATOR REQUESTS

DN, *n*	Set EQT device *n* down.
EQ, *n*	Print EQT entry *n*.
EQ, *n,p*	Delete (*p*=0) or specify (*p*=1) buffering.
GO, *name* [,*p1* [,*p2* [,.. [,*p5*]]]]]	Re-schedule suspended *name*.
IT, *name, R,MPT* [,*h, min* [,*s* [,*ms*]]]	Set time values of *name*.
LO [,*name* [,*lu*]]	Load absolute program into system.
LU ,*n* [,*m* [,*p*]]	Assign EQT *m* subchannel *p* to LU *n*, release *n* (*m*=0), or print *n* (m *absent*).
OF, *name, p*	Terminates *name* (*p*=0). Purge name (*p*=8). Abort *name* (*p* > 0).
ON, *name* [,*NOW*] [,*p1* ,*p2*,... *p5*]	Schedules *name*. *NOW* means ignore time values.
PL [,*lu*]	List ID segment information.
PR, *name, n*	Set priority of *name=n*.
RP, *name* [,*lu*]	Replace an absolute program.
SS, *name*	Suspend *name*.
ST, *name*	Print status of *name*.
TI	Print time.
TM, *day, h, min, s*	Specify day of year, and 24 hour time.
TO, *n* [,*m*]	Assign time-out value *m* to EQT *n*, or print value of *n* (*m* absent).
UP, *n*	Set EQT device *n* up.

APPENDIX F
SUMMARY OF EXEC CALLS

Consult Section III for complete details on each EXEC call.

ASSEMBLY LANGUAGE

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 parameters; must be
                        direct address

        DEF  p1    }
         .         }    Define addresses of parameters
         .         }    which may occur anywhere in
         .         }    program; may be multi-level
        DEF  pn    }    indirect

        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 Assembler Language calling sequence.

FORTRAN LANGUAGE

The general format on an EXEC call in FORTRAN is:

CALL EXEC (*p1 ..., pn*)

Where:

p1 through *pn* are either integer values or integer variables defined elsewhere in the program.

Example

```
CALL EXEC(7)
   or                  }
ICODE = 7              }  Equivalent calling sequences
CALL EXEC (ICODE)      }
```

Some EXEC call functions are handled automatically by the FORTRAN Compilers or special subroutines. Refer to Section III.

READ/WRITE

ICODE	DEC	1 (READ) or 2 (WRITE)
ICNWD	OCT	See Section III for control information.
IBUFR	BSS	*n* (*n*-word buffer)
IBUFL	DEC	*n* or *-2n* buffer length, words (+), characters (-)
IPRM1	DEC	*p* optional parameter.
IPRM2	DEC	*q* optional parameter.

I/O CONTROL

ICODE	DEC 3	
ICNWD	OCT	See Section III for control information.
IPRAM	DEC	*n* (Optional parameter required by some CONWDs)

I/O STATUS

ICODE	DEC 13	
ICNWD	DEC *n*	Logical unit number
ISTA1	NOP	Word 5 of EQT entry returned here
ISTA2	NOP	Optional parameter for word 4 of EQT

PROGRAM COMPLETION

ICODE	DEC 6

PROGRAM SUSPEND

ICODE	DEC 7

PROGRAM SCHEDULE

ICODE	DEC 9 or 10	9 (waiting) or 10 (no waiting)
INAME	ASC 3, *xxxxx*	*xxxxx* is the program name
p1	. . .	Up to five optional parameters.
p2	.	
	.	
	.	
p5	. . .	

TIME REQUEST

ICODE	DEC 11	
ITIME	BSS 5	Time values: tens of milliseconds, seconds, minutes, hours, day, returned in that order.

EXECUTION TIME
Initial Offset Version

ICODE	DEC 12	
IPROG	DEC 10	Schedule calling program, or
	ASC 3, *xxxxx*	Schedule *xxxxx*

(Continued)

EXECUTION TIME (Continued)

IRESL	DEC *x*	Resolution code (1=10's/ms; 2=secs; 3=mins; 4=hrs)
MTPLE	DEC *y*	Execution multiple
IOFST	DEC *-z*	*z* (units set by *x*) equals the initial offset.

EXECUTION TIME
Absolute Start-Time Version

ICODE	DEC 12	
IPROG	DEC 0	Schedule calling program, or
	ASC 3, *xxxxx*	Schedule *xxxxx*
IRESL	DEC *x*	Resolution code (1=10's/ms; 2=secs; 3=mins; 4=hrs)
MTPLE	DEC *y*	Execution multiple
IHRS	DEC *a*	Defines absolute start-time
MINS	DEC *b*	
ISECS	DEC *c*	
MSECS	DEC *d*	

APPENDIX G
LINE PRINTER FORMATTING

When a user program makes a READ/WRITE EXEC call to an HP line printer, the line printer driver, DVR12, interprets the first character in the line as a carriage control character and prints it as a space. †The control characters have the following meanings:

Character	Meaning
blank	Single space (print on every line)
0	Double space (print on every other line)
1	Eject page
*	Suppress space (overprint current line)
others	Single space.

Each printed line is followed by a single space unless suppressed by the control character asterisk (*). Double spacing requires an additional single space prior to printing the next line. If the last line of a page is printed and the following line contains a "1", then a completely blank page occurs.

When a user program makes an EXEC call for I/O control with the function bits in ICNWD set to octal 011 (see Section III, the optional parameter IPRAM word defines the format action to be performed by the line printer.

Parameter Word (Decimal)	Meaning
0	Page eject
0	Suppress space on the next print operation only
1 to 55	Space 1 to 55 lines ignoring page boundaries
56 to 63	Use carriage control channel equal to the word – 55
64	Set automatic page eject mode
65	Clear automatic page eject mode

CARRIAGE CONTROL CHANNELS

If the parameter word is between 56 to 63, 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.

† DVR12 checks for certain program names (FTN, ALGOL, ASMB, EDIT); for these programs it prints the first character of each line and generates a single space. As a result, the user must not specify automatic buffering on the line printer, since the program names are lost when the device is buffered.

INDEX

R

S

T

Q

SPECIAL CHARACTERS

PART NO. 29101-93001
Printed in U.S.A. 01/76

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
11000 Wolfe Road, Cupertino, California 95014