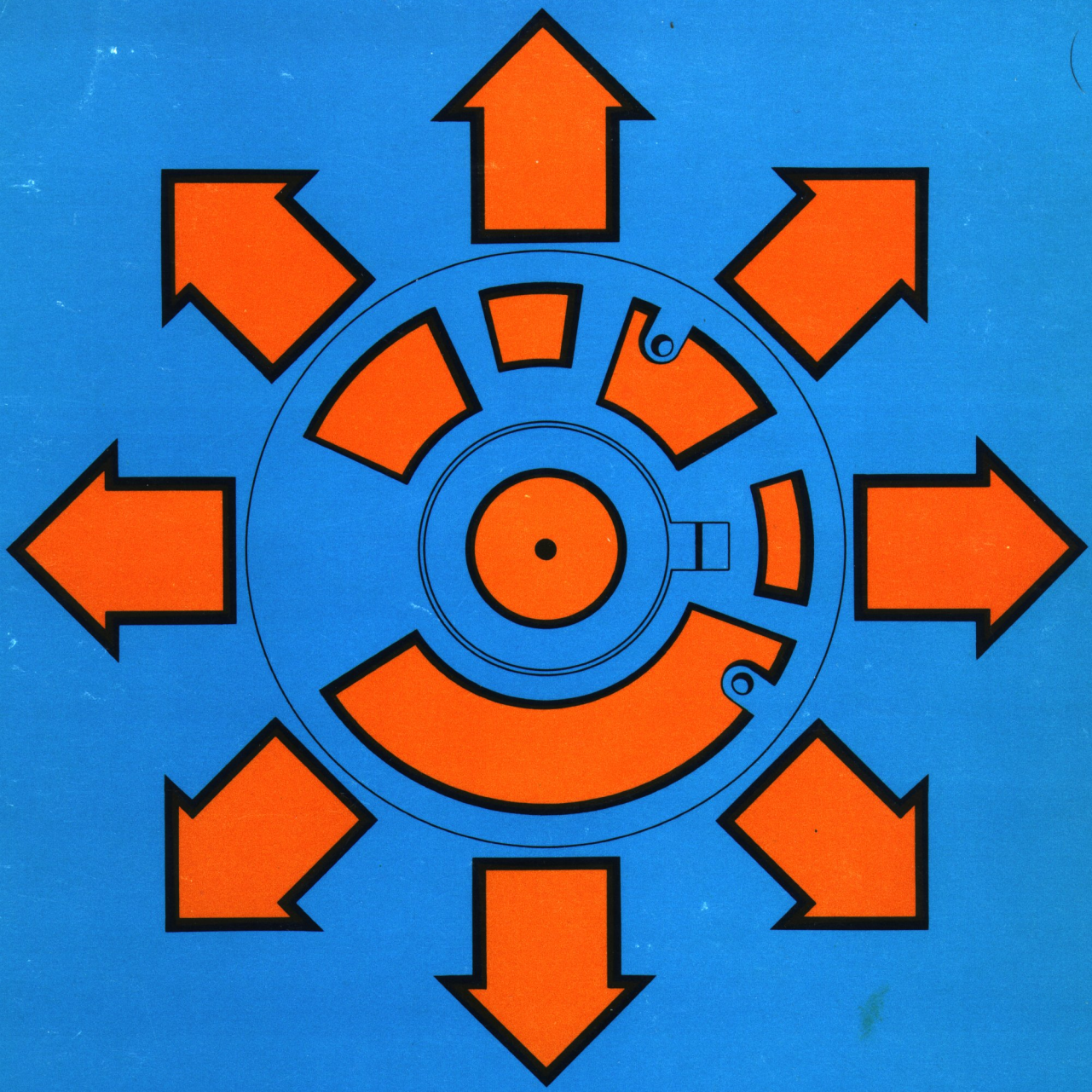


REAL-TIME EXECUTIVE-II SOFTWARE SYSTEM

PROGRAMMING AND OPERATING MANUAL

HEWLETT PACKARD

PROGRAMMING AND OPERATING MANUAL

REAL-TIME EXECUTIVE II

SOFTWARE SYSTEM

--IMPORTANT NOTICE--

This manual contains information on Hewlett-Packard Real-Time Executive Software. The reader is assumed to be a programmer familiar with one of the Hewlett-Packard programming languages, ALGOL, Assembler or FORTRAN.

PRINTED August 1975

Part No. 92001-93001

Microfiche No. 92001-93002

HEWLETT PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, U.S.A.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

TABLE OF CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

TABLES

GLOSSARY OF TERMS USED IN THIS MANUAL

ABSOLUTE SYSTEM – The absolute binary code of the Real-Time Executive II System (stored on logical unit 2).

AUXILIARY DISC – The disc is optional and when used is assigned to logical unit 3. (The absolute binary code of RTE-II does not reside on the auxiliary disc.) When the auxiliary disc is part of a moving head disc drive, it is contained on one subchannel of that drive. The auxiliary disc has the same status in the RTE-II as does the system disc in that it is treated as a logical extension of the system disc.

CONTROLLER – Two computer interface cards plugged into adjacent I/O slots, and connected to the disc drive with a cable.

DEVICE DOWN – (adj.) Relates to the state of a peripheral device. When the device is down, it is no longer operable. Also (noun), refers to the operator command DN, which sets the device down.

DEVICE UP – (adj.) Relates to the state of a peripheral device. When the device is up, it is operable. Also (noun), refers to the operator command UP, which sets the device up after it has been set down.

DISC DRIVE – Consists of a mechanism to rotate the disc, and electronic circuitry to write data on and read data off the disc through disc heads.

FIXED HEAD DISC DRIVE – Consists of the mechanism to rotate one disc with a read/write head per track.

GLOBAL TRACKS – Global tracks are a subset of RTE-II System tracks and are accounted for in the track assignment table. Any program can read/write or release a global track.

MOVING HEAD DISC DRIVE – Consists of a mechanism to rotate one or two discs, one permanently mounted and the other removable. There is one head per recording surface that is attached to a movable arm. The head is moved to the addressed track by means of an actuator driving the arm and head.

PERIPHERAL DISC – A peripheral disc is a disc that is available to the user for read/write operations but for which the Real-Time Executive II does not manage the disc nor maintain a track assignment table. A peripehral disc must have a logical unit number assignment greater than 6.

PROGRAM SWAPPING – Where program A is removed from computer memory and stored on the disc in its current state of execution, and program B is placed in the computer memory area (for execution) formerly occupied by program A. Program A is eventually returned to memory and continued.

REAL-TIME EXECUTIVE II – The total operating system comprised of three program modules, EXEC, SCHED, and RTIOC, plus I/O drivers, and various tables. Abbreviated RTE-II.

RTE-II SYSTEM TRACKS – All those disc tracks assigned to the system for which the RTE-II maintains a contiguous track assignment table. These disc tracks are located on logical unit 2 (system), and 3 (auxiliary).

SCRATCH AREA – A number of disc tracks used during system generation for storage of the relocatable binary code of RTE-II.

SUBCHANNEL – One of a group of peripherals connected to a single controller. For example, one moving head disc drive has two discs. The permanently mounted disc is subchannel 0 and the removable disc is subchannel 1.

SYSTEM DISC – The disc assigned to logical unit 2. When the system disc is part of a moving head disc drive, it is contained on one subchannel of that drive. The absolute binary code of the Real-Time Executive II resides on the system disc.

TIME-OUT – (adj.) Relating to the state of a peripheral device. When the device has timed-out, it is no longer operable. Also (noun), the parameter itself. Amount of time the RTE-II will wait for the device to respond to an I/O transfer command before RTE-II makes the device inoperable.

SECTION I
GENERAL DESCRIPTION

INTRODUCTION

The Real-Time Software Manual is a programmer's guide to using the Hewlett-Packard Real-Time Executive II Disc Based System (RTE-II). The reader should be familiar with operating procedures of the HP 2100 Series Computer and other system related hardware, and in addition, to software programming languages as presented in the FORTRAN (02116-9015), FORTRAN IV (5951-1321), ALGOL (02116-9072), and Assembler (02116-9014) Programmer's Reference Manuals.

HARDWARE

The HP Real-Time Executive II Disc Based System is divided as shown in Table 1.1.

Table 1-1. Minimum RTE-II System

HP 2100 Series Computer with 16K Memory and Memory Protect feature
Time Base Generator
Direct Memory Access
Disc Memory Subsystem
System Console Device
Paper Tape Reader
Tape Punch (Optional)
Line Printer (Optional)
Magnetic Tape (Optional)
HP 2313B Data Acquisition Subsystem (Optional)

SOFTWARE

Table 1-2 is a list of the software that makes up the Real-Time Executive II System. This list is comprised of the RTE-II software only and does not include System Input/Output (SIO) Drivers for the various peripheral equipment, or other hardware related drivers.

Table 1-2. Real-Time Software

Binary Tape Number	Description
92060-12004	Assembler
92002-16010	Interactive Editor EDITR
92001-16002	Relocating Loader
92001-16003	Multi-Terminal Monitor
92001-16004	Power Fail (DVR43)
92001-16005	RTE-II System Library
92001-16012	RTE-II Core Resident System
92001-16013	M.H. Generator
92001-16018	F.H. Generator
20747-60001	F.H. Disc Driver DVR30
20802-60001	System Dump
20805-60001	Editor
20875-60001/5	FORTRAN II
24016-60001	Prepare Tape System
24129-60001/2	ALGOL
24151-60001	RTE/DOS Relocatable Library (EAU)
24152-60001	RTE/DOS FORTRAN IV Library
24153-60001	RTE/DOS FORTRAN Formatter
24170-60001	RTE/DOS FORTRAN IV Compiler
24177-6001/3	RTE/DOS FORTRAN IV Compiler (Fast)
24248-60001	Relocatable Library
29013-60001	M.H. Disc Driver DVR31
29029-60001	Multi-Terminal Driver DVR00

SYSTEM DESCRIPTION

MULTIPROGRAMMING

The RTE-II System is a multiprogramming system that allows several programs to operate concurrently, each program executing during the unused central processor time of the others. All input/output and interrupt processing is controlled by RTE-II, except for special privileged interrupts which circumvent RTE-II for quicker response. When

a program requests a non-buffered I/O transfer, RTE-II 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-II reschedules the suspended program for execution. Operating programs can be written in Real-Time Assembler, ALGOL, or FORTRAN languages. Programs are scheduled by time intervals, an external device, an operator request, or by another program. RTE-II has a scheduling module which decides when to execute the competing programs.

The RTE-II system has up to four user defined program areas for execution of his programs.

- Real-time core resident
- Real-time disc resident
- Background core resident
- Background disc resident

Figure 1-1 shows the core layout for the core-resident and disc-resident programs.

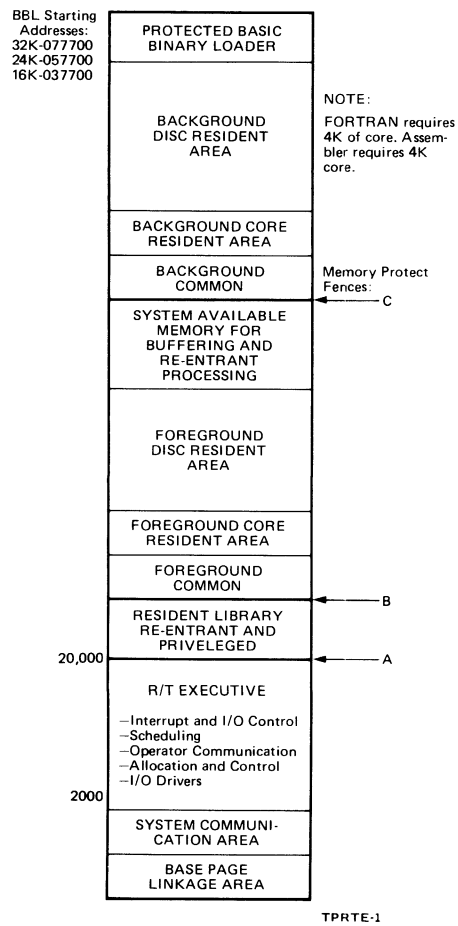


Figure 1-1. Core Allocations in the RTE-II System

REAL–TIME CORE RESIDENT

Real-time core resident programs are always resident in core and are intended for high priority tasks requiring quick response to real-time conditions. Programs of this type can control a logically subordinate set of disc resident programs.

REAL-TIME DISC RESIDENT

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

BACKGROUND CORE RESIDENT

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

BACKGROUND DISC RESIDENT

Just like the real-time disc resident type, background disc resident programs may also be swapped. The background area is usually used by compilers, assemblers, editors, etc. to create new programs to control future processes while the real-time area is handling external events. These new programs can be compiled, tested, and placed into operation without any paper tape output, and little operator intervention. In addition, new programs can be added to the set of permanent user programs, and the old ones deleted if desired. Background disc resident software usually includes the following items. Note that the memory requirements stated apply to the program plus a reasonable area for symbol tables.

REAL-TIME ASSEMBLER – Accepts storage programs in HP Assembler Language and outputs a relocatable binary program to a load-and-go disc track and/or punches the program on paper tape. Usually only one pass of the source tape is required.

REAL-TIME FORTRAN – Compiles FORTRAN programs and provides additional statements for real-time control. Same output options as Assembler.

REAL-TIME FORTRAN IV – Compiles source programs written in FORTRAN IV with extended precision arithmetic and bit manipulation via logical expressions. The 24170 version of FORTRAN IV compiler requires 6K of background core, and the 24177 version 12K.

REAL-TIME ALGOL – Compiles source programs written in HP ALGOL. An 8K background area is required to use this compiler with the real-time system.

REAL-TIME SYMBOLIC EDITOR – Edits, updates, and lists source language programs from a paper tape or disc file input.

REAL-TIME INTERACTIVE EDITOR (EDITR) – EDITR is an on-line editor that provides the user with a powerful editing tool (requires optional Batch Spool Monitor software). The user moves a pointer (pending line to be edited) through the file, forward or backward skipping lines if desired, and editing only where desired.

REAL-TIME RELOCATING LOADER – Provides on-line loading of user generated programs. Programs can be debugged and tested in background then brought up into real-time disc resident area if desired.

REAL-TIME DEBUG – When user program is loaded with relocating laoder, DEBUG can be appended to each main program and segments to provide checkout. When the program is run, DEBUG takes control of program execution and requests instructions from the keyboard.

SUBROUTINES

Each user program (main or segment) consists of a primary routine, containing the transfer point for entry into the program from RTE-II 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 Section IV, Part 6.) 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

The classification of a specific routine is based on its functions, word length, and execution times. A re-entrant or privileged subroutine may be used by more than one program at a time if they are in the resident library. (See Figure 1-1.) If called by disc resident programs and not in the resident library, the re-entrant and privileged subroutines are appended to the absolute version of the calling program.

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

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-II System can be generated such that one program is automatically scheduled each time the system is loaded from the disc. 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 RTGEN or on-line loading, and may be changed by an operator request.

The RTE-II 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 disc allocation
- 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

TIME SCHEDULING

Current time is updated every ten milliseconds. Whenever this occurs, a time list of programs is checked. Any programs scheduled to execute at that time are placed in the scheduled list. Programs can be executed on resolutions of hours, minutes, seconds, or even multiples of tens of milliseconds.

PRIORITY LEVEL

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 32767 (the lowest). The priority of any program can be changed by an operator request, and more than one program can be at the same priority.

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

- Scheduled
- Suspended

The base page communication area (see Appendix A) contains the pointers to the ID segment of the first, or highest priority, program in each list. Then, the linkage field of each ID segment contains the location of the next ID segment in the list. There is one scheduled list and five types of suspension lists:

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

PROGRAM INITIATION AND SWAPPING

A program is initiated immediately once it is transferred to core memory. Real-time resident programs are normally given highest priority and are always in core. If the program is on disc and is to be executed in one of the disc-resident areas, and an uncompleted program already occupies that space, the uncompleted program is transferred out to the disc and saved in its uncompleted and modified state. Then the new program is transferred into core. During the transfer a check is made to see if work can be done in one of the other areas of core. I/O operations continue simultaneously.

Background swapping makes it possible for multiple users to take advantage of the program development facilities of the RTE-II system. For example, one person can be editing his program while another is entering, compiling, assembling, or loading another program.

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-II System in detail, especially I/O drivers, and privileged interrupt processing.

INTERRUPT PROCESSING

All interrupts, except privileged interrupts and power fail, 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 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 and power fail, this instruction is a jump subroutine (indirect) to CIC.

PRIVILEGED INTERRUPT

RTE-II offers a special privileged interrupt feature, using an optional privileged interrupt control card and the hardware

priority structure of HP 2100 series Computers. Privileged interrupt bypasses normal interrupt processing to achieve faster response for interrupts having the greatest urgency.

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 of 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.

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 buffered. 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.

CLASS INPUT/OUTPUT OPERATIONS

All I/O operations of the system are performed concurrently with program computation. This is accomplished through a unique scheme of programming within the RTE-II system to effectively handle several programs addressing either other programs or I/O devices. The term "class" as used in this context is likened to an account which is owned by a program which may be used by a group of programs. The maximum number of classes is established during system generation. Once the numbers are established the system keeps track of them and assigns them (if available) to the calling program when a class I/O call is made. Once the number has been allocated, the user can keep it as long as desired and use it to make multiple class I/O calls. When the user is finished with the number it can be returned to the system for use by some other class user.

LOGICAL UNIT LOCK

The RTE-II system provides for temporary exclusive assignment of I/O devices to specific user's programs. This can be used to assure that a low-priority program completes its use of a printer, for example, without having that use preempted by a higher-priority program.

RESOURCE MANAGEMENT

Within RTE-II, any element that can be accessed by a user's program is regarded as a resource. A resource thus can be an I/O device, file, program, or subroutine. Occasionally, the user may want to manage a specific resource shared by a particular set of programs so that no two of these programs can use the resource at the same time. To accomplish this type of resource management the programs involved must mutually cooperate. For example, PROGB must not access a particular file when PROGA is using it. Both programs should include provisions for a hand-shaking arrangement overseen by the system when these programs are being executed concurrently. Under this arrangement, when PROGA has exclusive access to the file and PROGB attempts to access the same file, this access will be denied. PROGB will be suspended until PROGA releases its exclusive access. Then, PROGB can resume execution and access the file. (It is important to realize that as long as PROGB is suspended, it not only cannot access the file — it cannot perform any operations.) For more information refer to Section III.

The hand-shaking arrangement between programs is based upon an arbitrary resource identification number (RN) made available to programs. Within the cooperating programs the RN is related to a particular resource through the structure of the statements making up each program. When a program seeks exclusive access to a resource, it requests the system to lock the related RN. (This request is granted only if no other program has already locked the RN; otherwise, the requesting process is suspended until the RN is released.) When it is finished with the resource, the program requests the system to unlock the RN so that other programs can lock it.

A RN is not a physical entity. Furthermore, it is not logically assigned to any resource. The association between a RN and a resource is accomplished only by the context of the statements within the program using the RN. The RN is always known to the system but its meaning (the resource with which it is associated) is not. For this reason, all cooperating programs must agree on what RN is associated with what resource.

Programs can lock more than one RN at a time. However, in doing so, the users must be careful to avoid the case where two suspended programs cannot be resumed because they are mutually blocked.

EXEC COMMUNICATION (EXEC)

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

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

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

OPERATOR REQUESTS

The operator retains ultimate control of the RTE-II System with requests entered through the teleprinter keyboard. (See Section II.) Operator requests, which are handled by SCHED, can interrupt RTE-II 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 programs.
- Turn swapping on/off in either disc resident area.
- Set up load-and-go operations and source files.
- Declare I/O devices up or down.
- Dynamically alter the logical I/O structure and buffering designations.
- Eliminate disc-resident programs from the system.
- Examine and dynamically alter an I/O device's time-out parameter.
- Release tracks assigned to dormant programs.
- Initialize the real-time clock and print the time.

SYSTEM CONFIGURATION

User real-time programs and background programs, system programs, library routines, and Real-Time Executive Modules are incorporated into a configured RTE-II System. The RTE-II 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 Generator (RTGEN), the relocatable software modules and user programs are converted into a configured real-time system in absolute binary format which is stored on a disc, usually the HP 7900 Moving Head Disc Drive. In operation the configured system is loaded into the computer from the Real-Time Disc Resident area of the disc. The remaining disc storage is dynamically allocated by the configured system to user programs or is utilized by the scheduler for swapping operations.

The HP 7900 Moving Head Disc Drive is a single unit that contains two discs; one permanently mounted, and the other housed in a removable cartridge. The drive is interfaced to the computer through a single plug-in controller occupying two I/O slots. It is possible to daisy-chain up to four drives to the same controller providing up to eight discs. Each disc is a subchannel, and is accessed through a logical unit reference number that is referenced back to the equipment table (EQT) entry number of the controller. Therefore, one controller, containing eight subchannels linked to eight logical unit numbers, can control up to eight discs. When the RTE-II moving head disc system is generated using RTGEN, the user designates areas, or tracks, of each disc that are available to RTE-II. The system is limited to these tracks, and as a result, it is possible for the user to create many different configurations of RTE-II Systems, all coresiding on the same moving head disc channel.

Additional information on system configuration is found in Section VI and Appendix B of this manual.

SYSTEM/AUXILIARY DISCS

The RTE-II System disc tracks are those for which RTE controls and maintains a contiguous track usage table.

These are logical units 2 (system), and 3 (auxiliary). The system disc tracks are used for swapping, and by the editor, assembler, and compilers for source, load-and-go, and scratch area. They may also be used by user programs for storage. The only differences between a system disc and an auxiliary disc are that the absolute code of RTE-II is stored on the system disc, and that the auxiliary disc is optional.

PERIPHERAL DISCS

Peripheral discs are not managed by the RTE-II System but can be managed with the optional File Manager. Track allocation and usage in this case are totally up to the user through the File Manager. Peripheral discs are distinguished from system discs by having logical unit numbers greater than 6.

RTE SYSTEM SUMMARY

The Hewlett-Packard Real-Time Executive-II Software System is a multiprogramming, foreground-background system with priority scheduling, interrupt handling, and program load-and-go 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 in the background area during times when foreground real-time activities permit. The same computer can also be used by the programming group for ongoing development work with RTE-II's background compilers for FORTRAN, FORTRAN IV, and ALGOL, and with the HP Assembler, Editor, and other auxiliaries. Programs can be added to the system on-line, and on a load-and-go basis (no intervening paper tapes). For system protection, new programs can be debugged in background while memory protect maintains the integrity of the real-time foreground area.

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 are assigned by the user during RTGEN or on-line loading, and may 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 process 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-II 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 RTGEN, 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-II System by operator requests entered through the teleprinter console. These operator requests can interrupt RTE to perform the functions described in Table 2-1.

COMMAND STRUCTURE

The operator gains the attention of RTE-II by pressing any key on the console. When RTE-II responds with an asterisk (*), the operator types any operator request (or command), 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-II is restarted, any parameters previously changed are restored to their original value set during RTGEN.

- 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 system input device is a CRT terminal, the last character can be deleted with the backspace key. To delete the entire line use RUBOUT. Note that line feed is supplied by the system. Each request must be completed with an end-of-record terminator (e.g., carriage return for the teleprinter and CRT).

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

Table 2-1. RTE-II Operator Commands

Command Format	Description
AB	Abort current BATCH program.
BL	Sets buffer limits.
BR	Sets a break flag in named program's ID segment.
DN	Declare I/O device unavailable.
EQ	Examine the status of any I/O device, and dynamically alter device buffering assignments.
FL	Buffer flush command used in conjunction with Multiple Terminal Monitor (MTM) only.
GO	Restart programs out of suspension.
IT	Sets time intervals for programs
LG	Allocate load-and-go area.
LS	SEt logical source pointer.
LU	Dynamically alter device logical unit assignments.
OF	Turn programs off.
ON	Turn programs on.
PR	Change the priority of programs.
RU	Start a program immediately.
RT	Release program's disc tracks.
SS	Suspend programs.

Table 2-1. RTE-II Operator Commands (Continued)

Command Format	Description
ST	Examine the status of programs.
SW	Allow / disallow foreground / backgound programs to swap.
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.
[,*item1* ,*item2* ,*item3*]	This indicates that exactly one *item* may be specified.
item 1 *item 2* *item 3*	This indicates that there is a choice of entries for the parameter, but one parameter must be specified.
... (row of dots)	This notation means "and so on."

AB

Purpose:

To abort the current File Manager Program running under batch.

Format

AB [,0 / ,1]

Where:

0	terminates and removes from the time list the current BATCH program that is executing, scheduled, or operator suspended. Terminates BATCH programs which are I/O, memory, or disc suspended the next time they are scheduled. Disc tracks are not released.
1	terminates immediately the BATCH program and removes it from the time list, and releases all disc tracks. If suspended for I/O, the device and channel are cleared by a CLC.

COMMENTS

When the File Manager is waiting on a program it is running (e.g., ASMB), the AB command aborts that program just like the

OF,*name* command.

If the File Manager is dormant, or non-existent in the system, the AB command will cause the error message ILLEGAL STATUS to be printed. If the File Manager is not dormant and is not running a program, this command is the same as

BR,FMGR

BL

Purpose:

To examine or modify current Buffer Limits.

Format:

BL [*lower limit, upper limit,*]

Where:

BL	alone displays upper and lower limits previously set.
lower limit	is the lower limit number.
upper limit	is the upper limit number.

COMMENTS

Setting upper and lower memory limits with this command can prevent an inoperative or slow I/O device from monopolizing available system memory. Each time a buffered I/O request is made (Class I/O requests are buffered), the system adds up all the buffered words in I/O requests queued to that EQT entry and compares the number to the upper limit set by this command (or during generation). If the sum is less than the upper limit the new buffered request is added to the queue. If the sum is larger than the upper limit the requesting program is suspended in the general wait (STATUS = 3) list. When a buffered I/O request completes, the system adds up the remaining words in I/O requests queued to that EQT entry and compares the number to the lower limit set by this command (or during generation). When the sum is less than the lower limit, any programs suspended for exceeding the buffer limits on this EQT are rescheduled.

BR

Purpose:

To set an attention flag in a program's ID segment.

Format:

BR,*name*

Where:

name	is the name of the program.

COMMENTS

The BR command allows an operator to interrupt a program while it is running. When the BR command is executed, RTE sets bit 12 in word 21 of the named program's ID segment. The user's program can call an HP furnished subfunction that will test this bit and then act accordingly. The calling sequence of the subfunction is:

I = IFBRK (DM)

where DM is a dummy parameter to make the call appear as a function (DM need not be supplied in Assembly Language calls). The returned value will be negative if the break flag is set and positive if it is not. If the flag is set it will be cleared by IFBRK.

DN

Purpose:

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

Format:

DN,*eqt*

Where:

eqt	is 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 command. The operator might set a device down because of equipment problems, tape change, etc.

EQ (status)

Purpose:

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

Format:

EQ,*eqt*

Where:

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

COMMENTS

The status 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

Note that if *eqt* is 0 it is a bit bucket, and the LU associated with it is also a bit bucket.

EQ (buffering)

Purpose:

To change the automatic output buffering designation for a particular I/O device.

Format:

EQ,*eqt* [,*UNbuffer* / ,*BUffer*]

Where:

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

UNbuffer deletes buffering.

BUffer specifies buffering.

COMMENTS

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

FL

Purpose:

To eliminate buffered output to an I/O device.

Format:

lu>FL

Where:

lu is the logical unit number of the interrupting terminal.

COMMENTS

The FLush command can only be used in conjunction with the Multiple Terminal Monitor (MTM), and can only be entered from a terminal other than the system terminal.

Other methods of clearing the buffer are using an EXEC call or a File Manager command as follows:

CALL EXEC (3,23*lu*)

*ON,FMGR
:CN,*lu*,23B

GO

Purpose:

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

Format:

GO,*name* [,*p1* [, . . . [,*p5*]]]]]

Where:

name is the name of a suspended program to be scheduled for execution.

p1 ... *p5* is a list of parameters to be passed to *name* only when *name* has suspended itself (see Suspend EXEC Call in Section III). The parameters are not required if *name* was suspended with the SS command.

COMMENTS

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

Parameters *p1* through *p5* can be entered in ASCII or numeric form. Octal numbers are designated by the "B" suffix and negative numbers by a leading minus sign. For example:

GO,name,FI,LE,31061B

Note that only two ASCII characters per parameter are accepted; if one is given, the second character is passed as a blank (blank = 40B). If the first parameter is ASCII "NO" then it must be repeated (the system interprets it as "NOW" in the GO command). For example:

GO,name,NO,NO,FI,3,4,5

is interpreted as shown below. NO (NOW) is not used except to push the parameters out.

NO
FI
3
4
5

After a program has suspended itself and is restarted with the GO command, 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 the program has not suspended itself, the B-Register is restored to its value before suspension and the parameters are ignored.

IT

Purpose:

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

Format:

IT,*name* [,*res*,*mpt* [*hr*,*min* [,*sec* [,*ms*]]]]

Where:

name is the name of the program.

res is the resolution code
1 – tens of milliseconds
2 – seconds
3 – minutes
4 – hours

mpt is a number from 0 to 4095 which is used with *res* to give the actual time interval for scheduling (see Comments).

hr hours
min minutes
sec seconds
ms tens of ms.

} sets an initial start time.

COMMENTS

The resolution code (*res*) is the units in time to be multiplied by the multiple execution interval value (*mpt*) to get the total time interval. Thus, if *res*=2 and *mpt*=100, *name* would be scheduled every 100 seconds. If *hr*,*min*,*sec* and *ms* are present, the first execution occurs at the initial

start time which these parameters specify. (Program must be initialized with ON command.) If the parameters are not present (e.g., IT,*name*), the program's time values are set to zero and the program is removed from the time list. The program can still be called by another program or started with the ON, *name*,NOW or RU command.

When the system is restarted from the disc, time values set by the IT command are lost, and the original time values set at original load time are reinstated.

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

LG

Purpose:

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

Format:

LG,*numb*

Where:

numb = 0 (zero) releases the allocated load-and-go area.

numb > 0 release currently allocated load-and-go tracks and then allocate *numb* contiguous tracks for a load-and-go area.

COMMENTS

The user must allocate enough tracks for storing binary object code before each load-and-go compilation or assembly. If not, the compiler or assembler aborts and a diagnostic is printed on the system console.

IO06– Load-and-go area not defined.
IO09– Overflow of load-and-go area.

An LG request should not be used while a compiler or the Assembler is using the load-and-go tracks. If done, this may result in the message

LGO IN USE

being printed on the system console, and no change in the current number of load-and-go tracks. In most cases, however, it results in an IO06 error.

LS

Purpose:

To designate the disc logical unit number and starting track number of an existing source file before operating on it with EDIT, EDITR, FTN, FTN4, ALGOL, ASMB.

Format:

LS,*disc lu, trk numb*

Where:

disc lu is the logical unit number of the disc containing the source file.
2 or 3 = system or auxiliary disc units.
0 = eliminate the current source file designation.

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

COMMENTS

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

For details on creating, updating, compiling, or assembling source files, see Section IV, Background Programming.

LU (assignment)

Purpose:

To print the EQT table entry number and device subchannel number associated with a logical unit number.

Format:

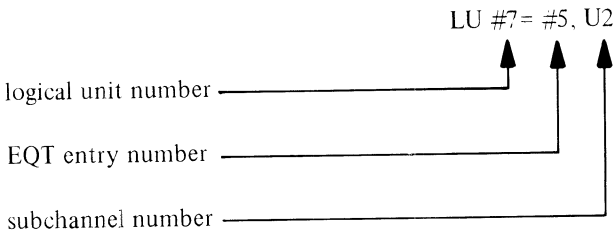
LU,*lu*

Where:

lu is a logical unit number from 1 to 63.

COMMENTS

The assignment information is printed as:



LU (reassignment)

Purpose:

To change a logical unit number assignment.

Format:

LU,*lu*,$^{eqt}_{0}$ [,*subch numb*]

Where:

lu	is a logical unit number from 1 to 63 (decimal).
eqt	is an EQT entry number to assign *lu*.
eqt	if zero (0) *lu* becomes the bit bucket.
subch numb	is a subchannel number (0 to 31) to assign to *lu*.

COMMENTS

The LU command can be used to change subchannel bits of DVR00 in reference to an EOT setting the tape reader down. Refer to the Multiple Device Driver DVR00 manual HP Part No. 29029-95001.

The restrictions on changing logical unit assignments are:

a. LU1 (system console) must be a keyboard entry device (e.g., teleprinter). Note that if LU1 is changed from one keyboard device to another, the new device will print a double asterisk (**).

b. LU2 (system disc) and LU3 (auxiliary disc) cannot be changed to another EQT entry number.

c. An LU cannot be changed to point at the same device as LU2 or LU3.

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 restarted from the disc, any assignments made by LU are reset to those originally set by RTGEN.

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 remove a disc resident program which was loaded on-line but not permanently incorporated into the protected RTE-II system.

Format:

OF, *name* [,0 / ,1 / ,8]

Where:

name	is the name of the program.
0	terminates and removes from the time list the named program the next time it is scheduled. The program's disc tracks are not released.
1	terminates immediately the named program, removes it from the time list, and releases all disc tracks. If suspended for I/O, the device and channel are cleared by a CLC.
8	terminates immediately the named program, and if the program is a temporary program loaded on-line, it is permanently removed from the system.

COMMENTS

For programs with segments, the OF, *name*, 8 command must be used on the segments as well as the main.

OF,*name*,8 will not remove permanent programs because their ID segments on the disc are not altered by this re-

quest. A permanent program is defined as a program loaded during generation, or on-line with the LOADR and with a copy of its ID segment in core and on the disc. For temporary programs loaded on-line the ID segment is blanked making the segment available for loading another program with LOADR. The tracks (if they are system tracks) containing the program are released. If the program had been stored on File Manager tracks, those tracks are not returned to the system but remain as File Manager tracks.

If the program is I/O suspended, the device and channel are cleared by a CLC. The OF,*name*,8 command must then be entered a second time to permanently remove *name* from the system in this case.

A permanent disc resident program is removed with the LOADR as described in Part 6 of Section IV.

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 system clock (see IT).
p1 ... *p5*	are parameters passed to the program when it is scheduled.

COMMENTS

Parameters *p1* through *p5* are the ones passed by RMPAR as described under Comments in the Program Schedule EXEC Call in Section III. Refer also to XTEMP words 2 through 6 in the program's ID segment (see Appendix A). Note that any parameters not entered as part of the ON command will be returned as zeros by a call to RMPAR.

Parameters *p1* through *p5* can be entered in ASCII or numeric form. Octal numbers are designated by the "B" suffix and negative numbers by a leading minus sign. For example:

ON,name,FI,LE,31061B

Note that only two ASCII characters per parameter are accepted; if only one is given, the second character is passed as a blank. (blank = 40B). If the first parameter is ASCII "NO" then it must be repeated (the system interprets it as "NOW" in the ON command). For example:

ON,name,NO,NO,FI,3,4,5

is interpreted as

NO
FI
3
4
5

If the resolution code in the ID segment of the program is not zero, RTE-II places the program in the time list for execution at specified times (unless *NOW* appears in which case, the program is scheduled and put into the time list immediately). 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 absolute start time or offset by some program in the system.

PR

Purpose:

To change the priority of a program.

Format:

PR,*name*,*numb*

Where:

name	is the name of the program.
numb	is the new priority.

COMMENTS

One (1) is the highest priority, and 32767 is the lowest. When the system is restarted from the disc, the priority of *name* resets to the value set by RTGEN or LOADR.

RU

Purpose:

To schedule a program immediately without affecting its entry in the time list. Up to five parameters may be passed to the program.

Format:

RU,*name*[,*p1*[, . . . [,*p5*]]]]]

Where:

name	is the name of a program.
p1 ... *p5*	are parameters passed to the program when it is scheduled.

COMMENTS

The RU command is usually used when the operator desires to run a program without affecting its entry in the time list.

Parameters *p1* through *p5* are the ones passed by RMPAR as described under Comments in the Program Schedule EXEC Call in Section III. Refer also to XTEMP words 2 through 6 in the program's ID segment (see Appendix A). Note that any parameters not entered as part of the RU command will be returned as zeros by a call to RMPAR.

Parameters *p1* through *p5* can be entered in ASCII or numeric form. Octal numbers are designated by the "B" suffix and negative numbers by a leading minus sign. For example:

RU,name,FI,LE,31061B

Note that only two ASCII characters per parameter are accepted; if only one is given, the second character is passed as a blank (blank = 40B). If the first parameter is ASCII "NO" then it must be repeated (the system interprets it as "NOW" in the RU command). For example:

RU,name,NO,NO,FI,3,4,5

is interpreted as shown below. NO(NOW) is not used except to push the parameters out.

NO
FI
3
4
5

RT

Purpose:

To release all disc tracks assigned to a program.

Format:

RT,*name*

Where:

name	is the name of the program that is to have its tracks released.

COMMENTS

If the program is not dormant, the command is illegal.

If the program is dormant, all tracks assigned to that program are released.

If any tracks are released as a result of this command, all programs in disc track allocation suspension are re-scheduled.

SS

Purpose:

To suspend a non-dormant program.

Format:

SS,*name*

Where:

name	is the name of the program to be suspended.

COMMENTS

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

The SS command is similar to the Program Suspend EXEC Call (see Section III).

ST

Purpose:

To request the status (priority, current list, time values) of a named program, or to determine the name of the foreground disc resident program, or background disc resident program currently occupying memory.

Format (status of a program):

ST,*name*

Format (name of current program):

ST [,0 / ,1 / ,2]

Where:

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

0 will cause the system to print the name of the program currently executing. If none, then 0 will be printed.

1 will cause the system to print the name of the current foreground disc resident program. If none, then 0 will be printed.

2 will cause the system to print the name of the current background disc resident program. If none, then 0 will be printed.

COMMENTS

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

pr s res mpt hr min sec ms T

Where

pr is the priority, a decimal value from 1 to 32767.

s is the current state of the program.
- 0– Dormant
- 1– Scheduled
- 2– I/O suspend
- 3– General wait
- 4– Unavailable memory suspend
- 5– Disc allocation suspend
- 6– Operator suspend or programmed suspend (EXEC 7 Call)
- 9– Background segment.

res, mpt, hr, min, sec 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 command).

A program is placed in the general wait list (status = 3) whenever:

a. It is waiting for a Resource Number (RN) to clear or become available. This includes Logical Unit (LU) locks and attempts to use a locked LU.
b. A schedule request is made with ICODE = 23 or 24 (queue schedule), and the program being called is busy.
c. A request is made to an I/O device that is down. This differs from a request to an I/O device that is busy.
d. A Class I/O GET Call is made and the Class Queue is empty.
e. A program is waiting for another program to complete as a result of an Exec 9 or 23 call.
f. A program is waiting on a Buffer limit (see BL this section).

Programs will be removed from the general wait list when the action waited for takes place, or when the program is aborted.

SW

Purpose:

To examine or change the foreground/background swapping word (base page 1736B).

Format:

SW [,0 ,1 ,2 ,3]

Where:

SW	alone displays swapping word.
0	indicates no swapping.
1	indicates foreground swapping only.
2	indicates background swapping only.
3	indicates both foreground and background swapping

COMMENTS

When SW with no parameters is entered, the swapping word is displayed as an octal number. The individual bits of the number are shown in Figure 2-1.

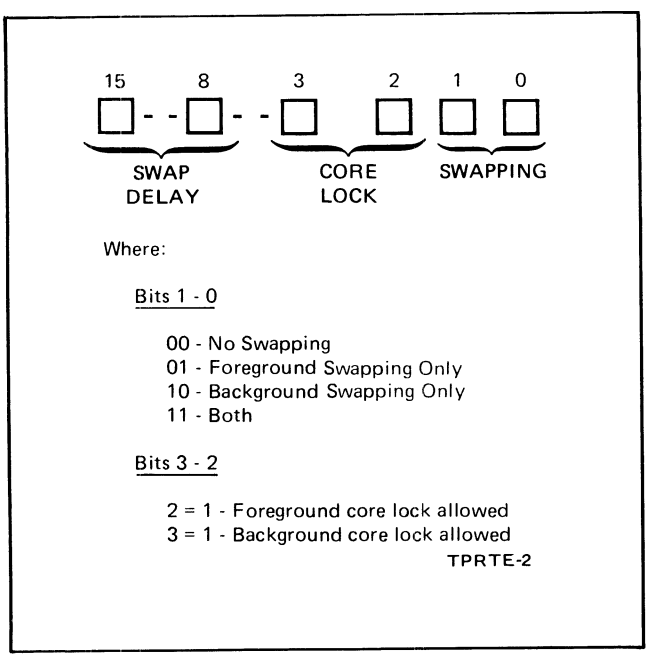


Figure 2-1. Swapping Word Display

Note that the core lock data shown in bits 2 and 3, and the swap delay time shown in bits 8 through 15 is for information only. The SW command will not modify these bits.

TI

Purpose:

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

Format:

TI

COMMENTS

The computer prints out the year, day and time:

YEAR DAY HR MIN SEC

Where

YEAR	is the four-digit year.
DAY	is the three-digit day of the year (see Table 2-3 for day of year conversion).
HR,MIN,SEC	is the time on a 24-hour clock.

The TI command is similar to the Time Request EXEC Call (see Section III).

TM

Purpose:

To set the real-time clock.

Format:

TM,*year*,*day*[,*hr*,*min*,*sec*]

Where:

year	is a four-digit year.
day	is a three-digit day of the year (see Table 2-3).
hr,*min*,*sec*	is the current time of a 24-hour clock.

Table 2-3. 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).

COMMENTS

The operator should give TM in response to the message printed when the RTE-II System is initiated from the disc:

SET TIME

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

NOTE

The real-time clock is automatically started from 8:00 on the approximate system release date 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,*eqt*[,*numb*]

Where:

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

numb is the number of 10 ms intervals to be used as the time-out value. (*numb* cannot be less than 500 (5 sec) for the system input device driven by DVR00/05).

COMMENTS

The time-out value is calculated using *numb* time-base generator interrupts (the time-base generator interrupts once every 10 ms). For example, *numb* = 100 sets a time-out value of one second: 100 · 10 ms = 1 second. When the system is restarted from the disc, time-out values set by TO are reset to the values originally set during RTGEN.

If *numb* is absent the time-out value of *eqt* is printed. The information is printed as:

TO #3 = 100

and means EQT entry number 3 has a time-out value of 100 ten millisecond intervals or one second.

If a device has been initiated, and it does not interrupt within the interval set by the time-out parameter, the following events take place:

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 I/O select code(s) through the EQT number located in the interrupt table.

UP

Purpose:

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

Format:

UP,*eqt*

Where:

eqt is the EQT entry number of the device to be re-enabled.

COMMENTS

When the operator of the RTE-II 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 command. 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). Previous requests made to this device are switched to the new device except in the case of buffered requests. To prevent indefinite I/O suspension on a downed device, time-out is used. Refer to I/O Device Time-Out in Section V, and the TO command in this section.

ERROR MESSAGES

RTE-II rejects operator requests for various reasons. When a request is in error, RTE-II prints one of the messages below. The operator should re-enter the request correctly.

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

Other errors may occur when an I/O device times out because of an inoperable state. When this occurs the operator can use the LU operator command to change the referenced device to another that works.

For example, the line printer may be in the OFF-LINE condition (or the operator has failed to engage the paper tape reader clutch). In this case the system will print one of the following error messages and suspend the program.

I/O ERR NR EQT *#eqt*
I/O ERR TO EQT *#eqt*

After the operator has corrected the device problem, all that is required is to type:

UP,*eqt*

where *eqt* is the downed device's equipment table entry number (same number given in the I/O error message). The program is automatically rescheduled and the desired I/O operation takes place.

Another example is that the program may be in the process of printing a long listing on the line printer when the printer runs out of paper. In this case it is possible to switch LU's and continue the listing without interruption as shown below.

I/O ERR TO EQT *#eqt*
LU,*lu*,*eqt*
UP,*eqt*

The error message says that the device at EQT number *eqt* has timed out and has been set down by the system. Note that some drivers handle time out themselves and do not cause the specified actions. In this case the error message states that the device is not ready (NR). The operator switches logical units (with the LU command) and then UP's the original equipment table entry that went down. The listing will continue on the new device.

SECTION III
EXEC CALLS

INTRODUCTION

This section describes the basic formats of FORTRAN, FORTRAN IV, ALGOL, 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 D at the rear of this manual for a summary of the EXEC calls and required parameters.

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 the EXEC module. EXEC 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 ALGOL, EXEC calls must be declared as CODE procedures and parameters must be declared as NAME. 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.

ERROR RETURN POINT

The user can alter the error return point of EXEC calls in association with error codes LU, SC, IO, DR, and RN 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 (LU,SC,LO,DR,RN) 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
```

Table 3-1. RTE-II EXEC Calls

Call	Request Code	Function	Page
READ,WRITE	1,2	Transfers information to and from an external I/O device.	3-4
Class I/O READ,WRITE WRITE/READ	17,18,20	Starts a no-wait I/O request which results in a transfer of information to and from an external I/O device or program.	3-6
I/O Control	3	Instigates various I/O control operations.	3-10
Class I/O Control	19	Instigates various I/O control operations under Class numbering scheme.	3-12
Class I/O Get	21	Completes the data transfer initiated by the Class I/O request.	3-13
I/O Status	13	Requests information about a device.	3-15
Disc Track Allocation		Assigns a specific number of disc tracks for data storage.	
Program	4		
Global	15		3-18
Disc Track Release		Release assigned disc tracks	
Program	5		
Global	16		3-19/3-20
Program Completion	6	Logically terminates execution of a calling program.	3-21
Program Suspend	7	Suspends calling program execution.	3-22
Program Segment Load	8	Loads a program segment into background area.	3-23
Program Schedule		Schedules a program for execution.	
	9	Immediate with wait.	
	10	Immediate without wait.	
	23	Queue with wait.	
	24	Queue without wait.	3-24
Time Request	11	Requests current real time.	3-26
Timed Execution			3-27
Initial Offset	12	Schedules a program for execution after an initial offset.	3-29
Absolute Start	12	Schedules a program for execution at a specified time.	
Resource Management	–	Allows cooperating programs a method of efficiently utilizing resources.	3-32
Program Swapping Control	22	Allows a program to lock itself into core and notify system of core usage.	3-31
Logical Unit Lock	–	Allows a program to exclusively dominate an I/O device.	3-34

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
```

ASSEMBLY LANGUAGE FORMAT

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

```
EXT       EXEC     Used to link
 .                  program to RTE-II.
 .
 .
JSB       EXEC     Transfer control to
                   RTE-II.
DEF       *+n+1    Defines point of
                   return from RTE-II;
                   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 (ICODE,p2...,pn)
```

Where

p1 through *pn* are either integer values or integer variables defined elsewhere in the program.

Example:

```
CALL EXEC (7)          }
      or               }   Equivalent calling
ICODE = 7              }   sequences
CALL EXEC (ICODE)      }
```

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

ALGOL FORMAT

In ALGOL certain conventions must be followed in making EXEC references (calls). The END statement in an ALGOL main procedure automatically declares the EXEC as external with a single integer parameter calling sequence. Therefore, without any further declaration of types of EXEC references through CODE procedures, any attempt to declare the EXEC with other than one integer parameter results in an error.

To avoid errors of this nature, CODE procedures with names other than EXEC must be declared with the proper number and type of parameters. A CODE procedure must be declared for each EXEC reference with a different number of type of parameters.

For example, the following is a main procedure referencing the EXEC with one parameter (EXECA), and then with four parameters (EXECB). The two respective CODE procedures are listed after the main procedure.

For example,

(In the main program):

```
:
PROCEDURE EXECA (A); INTEGER A; CODE;
PROCEDURE EXECB (A,B,C,D); INTEGER A,B,
C,D; CODE;
:
EXECA (I);
:
EXECB (J,K,L,M);
:
END$
```

```
(External)
   HPAL,P,B,L, "EXECA"
   PROCEDURE EXECA (A); INTEGER A;
   BEGIN PROCEDURE EXEC (A); INTEGER A;
   CODE; EXEC (A);
   END;
```

```
(External)
   HPAL,P,B,L, "EXECB""
   PROCEDURE EXECB (A,B,C,D); INTEGER A,B,
   C,D;
   BEGIN PROCEDURE EXEC (A,B,C,D);
   INTEGER A,B,C,D;CODE;EXEC (A,B,C,D);
   END;
```

READ/WRITE

Purpose:

To transfer information to or from an 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-II then reschedules the program.

Assembly Language:

```
                EXT     EXEC
                 :
                JSB     EXEC        Transfer control to RTE-II
                DEF     RTN         Return address
                DEF     ICODE       Request code
                DEF     ICNWD       Control information
                DEF     IBUFR       Buffer location
                DEF     IBUFL       Buffer length
                DEF     IPRM1       Optional parameter or track number if disc transfer
                DEF     IPRM2       Optional parameter or sector number if disc transfer
RTN             return  point       Continue execution (A = status, B = transmission log. If
                                    buffered WRITE, A and B are meaningless.)
                 :
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 or decimal track number if disc transfer
IPRM2           DEC     q           Optional parameter or decimal sector number if disc transfer
```

FORTRAN

```
DIMENSION IBUFR(n)      Set up buffer
IBUFL = n               Buffer length
ICODE = 2               Request code
ICNWD = conwd           Set Control Word
REG=EXEC (ICODE,ICNWD,IBUFR,IBUFL,IPRM1,IPRM2)
```

COMMENTS

Parameters IPRM1 and IPRM2 are optional, except in the case of disc transfers. If the data transfer involves a disc, IPRM1 is the disc track number and IPRM2 is the disc sector number. In calls to other I/O devices these parameters may have other uses. For example, driver DVR77 (HP 2323A Subsystem) uses IPRM1 for the scanner channel number and IPRM2 for the instrument program word. In some cases these parameters may be used to pass an additional control buffer to the driver (see Z-bit below).

CONTROL WORD

Figure 3-1 shows the format of the control word (*conwd*) required in the READ/WRITE calling sequence for DVR00 driven devices. Several fields defining the nature of the data transfer are shown.

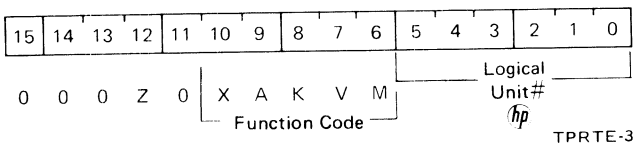


Figure 3-1. READ/WRITE (*conwd*) Format

Note that if the logical unit bits are specified as zero, the call takes place but no data is transferred.

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= 1 for the line printer will cause it to print column one.

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 (without printing) ASCII characters on the teleprinter (M = 0). (If A = 0, M determines mode of transfer.) This bit is effective on devices that recognize this control function.

X= 1 for moving head disc WRITE with cyclic checking, or 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).

Z= 1 designates that IPRM1 is the address of a control buffer and IPRM2 is the length of that buffer (only when the call is to a non-disc device). The length must be in words.

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

A- AND B- REGISTER RETURNS

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).

If the request is for output to a buffered device, the registers are meaningless.

I/O AND SWAPPING

Disc resident programs doing I/O are swappable under the following conditions:

a. The buffer is not in the disc resident area (i.e., it is in common or the resident library).

b. The device is buffered and the request is for output, and enough contiguous memory was allocated for buffering the record to be transferred.

Modified to contain request code before entry into driver.

c. The buffer is contained in the Temporary Data Block (TDB) of a re-entrant routine, and enough contiguous memory was allocated to hold the TDB.

Only the first buffer of a two buffer request (see Z-bit above) is checked to determine program swappability. It is the user's responsibility to put the second buffer in an area that implies the swappability if conditions "a" or "c" are true. The system takes care of case "b".

RE-ENTRANT I/O

A subroutine called REIO is furnished to allow the user to do re-entrant I/O. REIO is a utility type library routine and is more fully documented in Part 7 of Section IV, RTE-II Relocatable Libraries.

CLASS I/O – READ/WRITE

Purpose:

To transfer information to or from an external non-disc I/O device or another program. Depending on parameter options, the calling program will not be suspended while the call completes.

assembly Language:

```
            EXT   EXEC
             :
            JSB   EXEC      Transfer control to RTE-II
            DEF   RTN       Return address
            DEF   ICODE     Request code
            DEF   ICNWD     Control information
            DEF   IBUFR     Buffer location
            DEF   IBUFL     Buffer length
            DEF   IPRM1     Optional parameter
            DEF   IPRM2     Optional parameter
            DEF   ICLAS     Class word
RTN         return point    Contine execution (A = zero or status, B meaningless)
             :
ICODE       DEC   numb      17=READ, 18=WRITE, 20=WRITE/READ
ICNWD       OCT   conwd     conwd is described in Figure 3-1
IBUFR       BSS   n         Buffer of n words
IBUFL       DEC   n (or - 2n)  Same n; words (+) or characters (-)
IPRM1       DEC   f         Optional parameter (place holder)
IPRM2       DEC   q         Optional parameter (place holder)
ICLAS       OCT   class     class is described in Comments
```

FORTRAN:

```
            DIMENSION IBUFR (n)
            IBUFL = n
            ICODE = 20
            ICNWD = 0
            ICLAS = 0
            REG = EXEC (ICODE,ICNWD,IBUFR,IBUFL,IPRM1,IPRM2,ICLAS)
```

COMMENTS

Class I/O consists of a unique scheme of programming within the RTE-II system to effectively handle several programs addressing either other programs or I/O devices. The following description of class I/O relies upon a Glossary of Terms directly related to Class I/O (see Table 3-2).

The maximum number of classes is established during system generation after the last system modules are loaded. The generator requests how many class numbers are to be established and the operator responds with a number between 0 and 255. Once the numbers are established the system keeps track of them and assigns them (if available) to the calling program when a class I/O call is made and the Class Number parameter is set to zero. Once the number has been allocated, the user can keep it as long as desired and use it to make multiple class I/O Calls. When the user is finished with the number it can be returned to the system for use by some other class user. One example of using Class I/O is Class I/O Mailbox communication. The example program in Figure 3-3 and described in the following sequence of events, shows how this is accomplished.

Table 3-2. Glossary of Terms for Class Input/Output

Term	Description
1. Class	An account which is owned by a program which may be used by a group of programs.
2. Class Number	The account number referred to in number one.
3. Class Users	Programs that use the class number.
4. Class Request	An access to a logical unit number with a class number.
5. Class Members	Logical unit numbers that are currently being accessed in behalf of a class. Completion of access removes the association between class number and logical unit number (completion of access is defined as when the driver completes the request).
6. Class Queue (Pending)	The set of uncompleted class requests.
7. Class Queue (Completed)	The set of all completed class requests. The structure is first in, first out.

a. User program PROGA issues a Class I/O call with the Class Number parameter set to zero and the logical unit number portion of the control word parameter set to zero. This causes the system to allocate a Class Number (if available) and the request to complete immediately. (Logical unit zero specifies a system "bit bucket" which implies immediate completion).

b. When the WRITE/READ call completes, PROGA's data will have been placed in a system buffer and this fact recorded in the Completed Class Queue for this class.

c. PROGA then schedules PROGB (the program receiving the data) and passes PROGB, as a parameter, the Class Number it obtained.

d. When PROGB executes it picks up the Class Number by calling RMPAR. Then using this Class Number, it issues a Class I/O Get Call to the class. PROGA's data is then passed from the system buffer to PROGB's buffer.

The system handles a Class I/O call in the following manner.

a. When the class user issues a Class I/O call (and the call is received), the system allocates a buffer from available memory and puts the call in the header (first 8 words) of this buffer. The call is placed in the pending class queue and the system returns control to the class user.

b. If this is the only call pending on the EQT, the driver is called immediately, otherwise the system returns control to the class user and calls the driver according to program priority.

c. If buffer space is not available, the class user is memory suspended unless bit 15 ("no wait") is set. If the "no wait" bit is set, control is returned to the class user with the A-Register containing a −2 indicating no memory available.

d. If the class number is not available or the I/O device is down, the class user is placed in the general wait list (status = 3) until the condition changes.

e. If the call is successful, the A-Register will contain zero on return to the program.

The buffer area furnished by the system is filled with the caller's data if the request is either a WRITE, or a WRITE/READ call. The buffer is then queued (pending) on the specified logical unit number. Since the system forms a direct relationship between logical unit numbers and EQT entries, the buffer can also be thought of as being queued on the EQT entry.

After the driver receives the Class I/O call (in the form of a standard I/O call) and completes, the system will:

a. Release the buffer portion of the request if a WRITE. The header is retained for the Get call.
b. Queue the header portion of the buffer in the Completed Class Queue.
c. If a Get call is pending on the Class Number, reschedule the calling program. (This means that if the user issues a Class Get call or examines the completed Class Queue before the driver completes, the user has effectively beat the system to the completed Class Queue.) Note that the program that issued the Class I/O call and the program that issued the Class Get call do not have to be the same program.
d. If there is no Get call outstanding, the system continues and the driver is free for other calls.

When the user issues the Get call, the completed Class Queue is checked and one of the following paths is taken.

a. If the driver has completed, the header of the buffer is returned (plus the data). The user (calling program) has the option of leaving the I/O request in the completed Class Queue so as not to lose the data. In this case a subsequent Get call will obtain the same data. Or the user can dequeue the request and release the Class number.
b. If the driver has not yet completed (Get call beat system to the completed Class Queue), the calling program is suspended in the general wait list (status = 3) and a marker so stating is entered in the completed Class Queue header. If desired, the program can set the "no wait" bit to avoid suspension. In any case, when the driver completes, any program waiting in the general wait list for this class is automatically rescheduled. Note that only one program can be waiting for any given class at any instant. If a second program attempts a GET call before the first one has been satisfied it will be aborted (I/O error IO10).

IPRM1 and IPRM2 are required as place holders in this request. They may also be used to pass information through to the Class I/O Get Call to aid in processing the request.

For a combination class WRITE/READ call, the driver should expect control data in the buffer IBUFR. The system will treat the request as a class WRITE in that the buffer must be moved prior to the driver call, and as a class READ in that the buffer must be saved after the driver completion. Note that the driver will receive a standard READ request (ICODE = 1) on this request.

Refer to Figure 3-1 for the format of the control word (*conwd*) required in the class I/O READ/WRITE calling sequence.

Figure 3-2 shows the format of the class word (ICLAS) required in the calling sequence. To obtain a class number from the system the class portion (bits 12-0) of the word is set to zero. This causes the system to allocate a Class Number (if one is available) to the calling program. The number is returned in the ICLAS parameter when the call completes and the user must specify this parameter (unaltered) when using it for later calls. Bit 15 is the "no-wait" bit. When set the calling program does not memory suspend if memory (or a class number) is not available. A-Register value when the program returns is as follows:

"A" Value	Reason
0	OK-request done
−1	No class number
−2	No memory now or buffer limit exceeded.

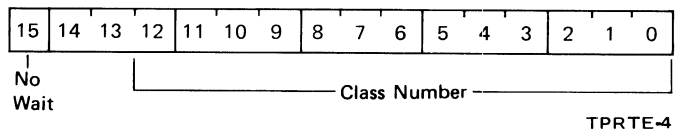


Figure 3-2. Class Number (ICLAS) Format

When the user's program issues a Class I/O call (and the call is received) the system allocates a buffer from available memory and puts the call in this buffer. The call is queued and the system returns control to the user's program. If memory is not available, three possible conditions exist: (1) The program is requesting more memory than will ever be available. In this case the program is aborted with a IO04 error. (2) The program is requesting a reasonable amount of memory but the system must wait until memory is returned before it can satisfy the calling program. In this case the program is suspended unless the "no wait" bit is set in which case a return is made with the A-Register set to −2. (3) If the buffer limit is exceeded the program will be suspended until this condition clears. If the "no wait" bit is set the program is not suspended and the A-Register is set to −2.

```
FTN,L
      PROGRAM PROGA
      DIMENSION IBFR(32),INAME(3)
            .
            .
            .
C
C     DO CLASS WRITE/READ TO LU=0.
C
      ICLAS=0
      CALL EXEC(20,0,IBFR,-64,IDUMY,JDUMY,ICLAS)
C
C     SCHEDULE RECEIVING PROGRAM AND PASS IT CLASS#.
C
      INAME(1)=501228
      INAME(2)=47507B
      INAME(3)=410008
      CALL EXEC(10,INAME,ICLAS)
            .
            .
            .
      END



FTN,L
      PROGRAM PROGB
      DIMENSION IBFR(32),IPRAM(5)
C
C     SAVE CLASS #, IPRAM(1)
C
      CALL RMPAR(IPRAM)
C
C     ACCEPT DATA FROM PROGA USING CLASS GET CALL
C     AND RELEASE THE CLASS NUMBER.
C
      CALL EXEC(21,IPRAM(1),IBFR,32)
            .
            .
            .
```

Figure 3-3. Example of Class I/O Mailbox Communication

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-II
        DEF   RTN      Return address
        DEF   ICODE    Request code
        DEF   ICNWD    Control information
        DEF   IPRAM    Optional parameter
RTN     return point   Continue execution (A = status, B meaningless. If call is
        :              buffered, A is meaningless)
ICODE   DEC   3        Request code = 3
ICNWD   OCT   conwd    See Control Word
IPRAM   DEC   n        Required for some control functions; see Control Word
```

FORTRAN:

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

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

CONTROL WORD

Figure 3-4 shows the format of the control word (*conwd*) required in the I/O control calling sequence.

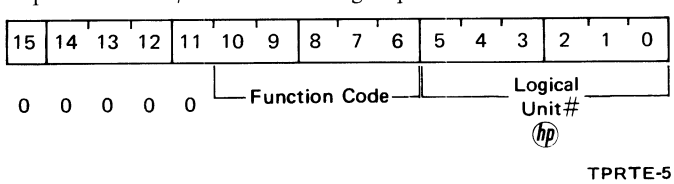


Figure 3-4 I/O Control (*conwd*) Format

Function Code (Octal)	Action
00	Unused
01	Write end-of-file (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 3-inch gap (magnetic tape)

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

Function Code (Octal)	Action
13	Forward space file (magnetic tape)
14	Backward space file (magnetic tape)
15	Conditional form feed (see Line Printer Driver manual).

The following functions are defined for DVR00. For more information see the driver manual 29029-60001.

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 parameter is set as the new time-out interval.

23 Ignore all further action requests until:

a) The device queue is empty or
b) An input request is encountered in the queue, 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 or the number of lines to be spaced on the teleprinter. A positive parameter always spaces that number of lines.

CLASS I/O – CONTROL

Purpose:

To carry out various I/O control operations, such as backspace, write end-of-file, rewind, etc. The calling program does not wait.

Assembly Language:

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      Transfer control to RTE-II
            DEF   RTN       Return address
            DEF   ICODE     Request code
            DEF   ICNWD     Control information
            DEF   IPRAM     Optional parameter
            DEF   ICLAS     Class word
RTN         return point    Continue execution (A = class number, B meaningless)
             .
             .
             .
ICODE       DEC   19        Request code = 19
ICNWD       OCT   conwd     See Control Word
IPRAM       DEC   n         Required for some control functions; see Control Word
ICLAS       OCT   class     class is described in Comments
```

FORTRAN:

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

```
ICODE = 19              Request code
ICNWD = conwd           See Control Word
IPRAM = x               See Control Word
ICLAS = y               Class Word
REG = EXEC (ICODE,ICNWD,IPRAM,ICLAS)
```

COMMENTS

Refer to Figure 3-4 for the format of the control word (*conwd*) required in the Class I/O control calling sequence.

Note that this call, with the exception of the ICLAS parameter is the same as the standard I/O control call. Also refer to the Class I/O Get Call for additional information.

CLASS I/O – GET

Purpose:

To complete the data transfer between the system and user program that was previously initiated by a Class request.

Assembly Language:

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC      Transfer control to RTE-II
                DEF   RTN       Return address
                DEF   ICODE     Request code
                DEF   ICLAS     Class word
                DEF   IBUFR     Buffer location
                DEF   IBUFL     Buffer length
                DEF   IRTN1     Optional parameter status word
                DEF   IRTN2     Optional parameter status word
                DEF   IRTN3     Optional parameter class word
RTN             return address  Continue execution (A = status, B = Transmission log)
                 .
                 .
                 .
ICODE           DEC   21        21 = class GET call
ICLAS           NOP             class is described in Comments
IBUFR           BSS   n         Buffer of n words
IBUFL           DEC   n (or -2n) Same n; words (+) or characters (-)
IRTN1           NOP             Location for IPRM1 from READ/WRITE call
IRTN2           NOP             Location for IPRM2 from READ/WRITE call
IRTN3           NOP             Location for IPRM3 from READ/WRITE call
```

FORTRAN:

```
DIMENSION IBUFR (n)
IBUFL = n
ICODE = 21
ICLAS = x0
REG = EXEC (ICODE,ICLAS,IBUFR,IBUFL,IRTN1,IRTN2,IRTN3)
```

COMMENTS

When the calling program issues a Class Get call, the program is telling the system that it is ready to accept returned data from a Class READ call or remove a completed Class WRITE or Control call from the completed Class list. If the driver has not yet completed (Get call beat system to the completed Class Queue), the calling program is suspended in the general wait list (status = 3) and a marker so stating is entered in the Class Queue header. When the driver completes, the program is automatically rescheduled. If desired, the program can set the "no wait" bit to avoid suspension.

Figure 3-5 shows the format of the class word (ICLAS) required in Class Get Call. Bits 12-0 represent the Class Number and security code that the Get call is looking for. This Class Number is obtained (in unaltered form) from the original Class I/O READ, WRITE, CONTROL or WRITE/READ call. Bit 15 is the "no wait" bit. When set, the calling program does not suspend if the Class Request has not yet completed. Bit 14 is the "save" bit. When set, the buffer is not released; therefore, a subsequent Get call will return the same data. Bit 13 is the "de-allocate" bit. When set, the Class Number is not returned to the system. If bit 13 is zero and no requests are left in the Pending Class Queue, and no Class Requests for this class are waiting for

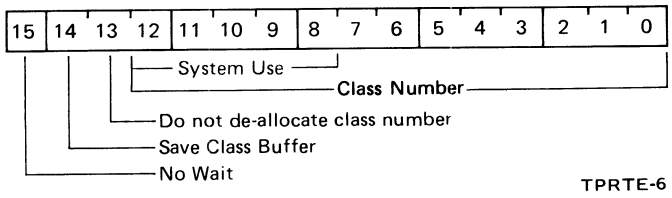


Figure 3-5. Class Word (ICLAS) Format

driver processing, the class is returned to the system. It is possible for the call to return the Class Number and data, or no data depending on if there is one class call left. Bits 14 and 13 work in conjunction with each other. If bit 14 is set then the buffer will not be released. Therefore you cannot de-allocate the Class Number. That is, the Class Number cannot be released because there is still an outstanding request against it.

Only when the Get call gets the last class request on a class, or on an empty class queue (completed and pending) can the user release the Class Number by clearing bit 13 in the ICLAS word.

Three parameters in the call are return locations: that is, values from the system are returned to the calling program in these locations. Optional parameters IPRM1 and IPRM2 from the Class I/O – WRITE/READ calls are returned in IRTN1 and IRTN2. These words are protected from modification by the driver. The original request code received by the driver is returned in IRTN3. For example:

Original Request Code	Value Returned in IRTN3
17/20(READ,WRITE/READ)	1
18 (WRITE)	2
19 (CONTROL)	3

BUFFER CONSIDERATIONS

Several buffer considerations exist in the Class I/O Get call. They are as follows:

a. The number of words returned to the user's buffer is the minimum of the requested number and the number in the Completed Class queue element being returned.

b. If the original request was made with the "Z" bit set in the control word, then IPRM1 returned by this call will be meaningless.

c. The "Z" buffer will be returned if there is room for it (see "a" above) only if the original request was a READ or WRITE/READ (i.e., for WRITE requests no data is returned in the buffer area).

A- AND B-REGISTER RETURNS

The A- and B-Registers are set as follows after a Class I/O Get call.

A-Register	B-Register
A15 = 0 then A = status	B = transmission log (positive words or characters depending on original request)

A15 = 1 then A = –(*numb+1*)B = meaningless

On return with data, bit 15 is set to zero and the rest of the A-Register contains the status word (EQT5). If a return is made without data (the "no wait bit" was set in the class word) then bit 15 is set to one and the A-Register contains the number of requests *numb* made to the class bit not yet serviced by the driver (i.e., pending class requests).

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-II
                DEF   RTN       Return address
                DEF   ICODE     Request code
                DEF   ICNWD     Control information
                DEF   IEQT5     Status word 1
                DEF   IEQT4     Status word 2 - optional
RTN             return point    Continue execution (A and B are meaningless)
                 .
                 .
                 .
ICODE           DEC   13        Request code = 13
ICNWD           DEC   n         Logical unit number
IEQT5           NOP             Word 5 of EQT entry returned here
IEQT4           NOP             Word 4 of EQT entry returned here, optional
```

FORTRAN:

```
ICODE = 13                      Request code
ICNWD = nn                      nn is the logical unit number
CALL EXEC (ICODE,ICNWD,IEQT5,IEQT4)
```

COMMENTS

When this call is made the calling program is not suspended. Equipment Table (EQT) words 5 and 4 (optional) are returned in IEQT5 and IEQT4 and are defined as shown in Table 3-3. The STATUS portion of EQT word 5 is further broken down and shown in Table 3-4.

Table 3-3. I/O Status Word (IEQT5/IEQT4) Format

WORD	CONTENTS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	D	B	P	S	T	Unit #					Channel #					
5	AV		EQUIP. TYPE CODE					STATUS (see Table 3-4)								

IEQT4

D = 1 if DMA required.

B = 1 if automatic output buffering used.

P = 1 if driver is to process power fail.

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).

IEQT5

AV = availability indicator:

0 = available for use.

1 = disabled (down).

2 = busy (currently in operation).

3 = waiting for an available DMA channel.

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

00 to 07_8 = paper tape devices (or system control devices)

00 = teleprinter (or system keyboard control device)

01 = photo-reader

02 = paper tape punch

05 = interactive keyboard device

10 to 17 = unit record devices

10 = plotter

12 = line printer

15 = mark sense card reader

20 to 37 = magnetic tape/mass storage devices

30 = fixed head disc or drum

31 = 7900 moving head disc

32 = 7905 moving head disc

40 to 77 = instruments

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.

Table 3-4. EQT Word 5, STATUS Table.

Device \ Status	7	6	5	4	3	2	1	0
Teleprinter(s) Photoreader(s) Punch(es) DVR00	X	—	End of I/O Tape	—	—	STL	TEN	
7210 Plotter DVR10	—	—	—	—	—	—	—	PD
2892 Card Reader DVR11	HE							IOL/CE
2767 Line Printer DVR12						"<" LCF	"*" LCF	NE
2607 Line Printer DVR12	TOF	DM	ON	RY	X	X	Auto page eject	X
7261 Card Reader 2761 Mark Sense Reader DVR15	EOF —	— —	HE/SF HE/SF	PF PF	— —	— —	DE DE	RNR RNR
3030 Mag Tape 7970 DVR22 DVR23	EOF	ST	EOT	TE	I/OR	NW	PE	DB/OL
2766 Fixed Head 2773 Disc/Drum DVR30	DR(1) NR(0)	—	SAC	—	AF	WE	PE	DB
7900 Moving Head Disc DVR31		NR	EOT	AE	FC	SC	DE	EE
7905 Moving Head Disc DVR32	PS	FS	HF	RY	SC	NR		EE

Where:

PE = Parity Error
HE = Hopper Empty
SF = Stacker Full
RNR = Reader Not Ready
PF = Pick Fail
DE = Data Error
OL = Off Line
ON = On Line
CE = Compare Error
BT = Broken Tape
DB = Device Busy
EOF = End of file
ST = Start of Tape
TE = Timing Error
I/OR = I/O Reject
NW = No Write (write enable ring missing or tape unit is rewinding)
SC = Seek Check
FC = Flagged Cylinder
AE = Address Error
EOT = End of Track
NR = Not Ready
RY = Ready (0 = power on)
LCF = Last Character Flag
NE = No Error
DR = Disc Ready
SAC = Sector Address Coincidence (troubleshooting only)
PS = Protect Switch Set
FS = Drive Format Switch is Set
HF = Hardware Fault
AF = Abort Flag [NR (Bit 7=0) has occurred during since last data transfer]
WE = Currently addressed track is write enabled.
EE = Error exists
TEN = Terminal Enabled
TOF = Top of Form
DM = Demand (1 = Idle)
X = Driver Internal Use
STL = Stall required in program
PD = Pen Down

DISC TRACK ALLOCATION

Purpose:

To request that RTE-II assign a specific number of contiguous disc tracks for data storage. The tracks are either assigned to the calling program or assigned globally.

Assembly Language:

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC     Transfer control to RTE-II
            DEF   RTN      Return address
            DEF   ICODE    Request code
            DEF   ITRAK    Number of contiguous tracks required
            DEF   ISTRK    Start track number
            DEF   IDISC    Disc logical unit number
            DEF   ISECT    Number of 64 word sectors/track
RTN         return point   Continue execution (A and B are meaningless)
             .
             .
             .
ICODE       DEC   4 or 15  4 = allocate track to program
                           15 = allocate track globally
ITRAK       DEC   n        n = number of contiguous tracks within the same disc unit
                           requested. If bit 15 of ITRAK = 1 the program is not suspended if
                           tracks are not available; if bit 15 = 0, the program is suspended until
                           the tracks are available.
ISTRK       NOP            RTE-II stores starting track number here, or -1 if the tracks are not
                           available.
IDISC       NOP            RTE-II stores logical unit number here.
ISECT       NOP            RTE-II stores number of 64 word sectors/track here.
```

FORTRAN:

Example (with no suspension):

```
ICODE = 4
ITRAK = 100000B + n
CALL EXEC (ICODE,ITRAK,ISTRK,IDISC,ISECT)
```

Example (with suspension until tracks available):

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

COMMENTS

RTE-II supplies only whole tracks within one disc. When writing or reading from the tracks (see READ/WRITE EXEC Call), RTE-II does not provide automatic track switching; the user program (when using this call) is completely responsible for file and track management. RTE-II will prevent other programs from writing on program assigned tracks, but not from reading out of them.

The program retains the tracks until it or the operator releases them, or the program is aborted.

Globally assigned tracks are available to any program for READ, WRITE, or release. The user is completely responsible for their management. RTE-II will not prevent other programs from writing on globally assigned tracks or releasing them.

DISC TRACK RELEASE-PROGRAM TRACKS

Purpose:

To release some contiguous disc tracks which were previously assigned to a program (see Disc Allocation EXEC Call).

Assembly Language:

```
        EXT   EXEC
        .
        .
        .
        JSB   EXEC     Transfer control to RTE-II
        DEF   RTN      Return address
        DEF   ICODE    Request code
        DEF   ITRAK    Number of contiguous tracks, or -1
        DEF   ISTRK    Starting track number
        DEF   IDISC    Disc logical unit
RTN     return point   Continue execution (A and B are meaningless)
        .
        .
        .
ICODE   DEC   5        Release program's tracks
ITRAK   DEC   n        If n = -1, release all tracks assigned to program; ISTRK and IDISC are
                       unnecessary. Otherwise, n is the number of contiguous tracks to be
                       released starting at ISTRK.
ISTRK   DEC   m        Starting track number
IDISC   DEC   p        Disc logical unit
```

FORTRAN:

Release of *n* contiguous tracks starting at *m* on LU *p*:

```
ICODE = 5
ITRAK = n
ISTRK = m
IDISC = p
CALL EXEC (ICODE,ITRAK,ISTRK,IDISC)
```

Release all tracks allocated to the program.

```
ICODE = 5
ITRAK = -1
CALL EXEC (ICODE,ITRAK)
```

COMMENTS

When tracks are released, any program suspended waiting for tracks is rescheduled.

DISC TRACK RELEASE-GLOBAL TRACKS

Purpose:

To release some contiguous disc tracks which were previously assigned globally (see Disc Allocation EXEC Call).

Assembly Language:

```
        EXT   EXEC
        :
        JSB   EXEC     Transfer control to RTE-II
        DEF   RTN      Return address
        DEF   ICODE    Request code
        DEF   ITRAK    Number of contiguous tracks
        DEF   ISTRK    Starting track number
        DEF   IDISC    Disc logical unit
RTN     return point   Continue execution (A = track release status, B meaningless)
        :
ICODE   DEC   16       Release global tracks
ITRAK   DEC   n        The number of contiguous tracks to be released starting at ISTRK
ISTRK   DEC   m        Starting track number
IDISC   DEC   p        Disc logical unit
```

FORTRAN:

Release of *n* contiguous global tracks starting at *m* on LU *p*:

```
ICODE = 16
ITRAK = n
ISTRK = m
IDISC = p
REG = EXEC (ICODE,ITRAK,ISTRK,IDISC)
```

COMMENTS

If any one of the tracks to be released is either not assigned globally or is currently in use (i.e., some program is queued to read or write on the track at the time of the release request), none of the tracks are released.

The requesting program is rescheduled after the request with the A-Register set as follows:

A= 0	The tracks have been released.
A= −1	No tracks have been released – at least one of them was in use.
A= −2	No tracks have been released – one or more of them was not assigned globally.

PROGRAM COMPLETION

Purpose:

To notify RTE-III that the calling program wishes to instigate a program termination.

Assembly Language:

```
            EXT     EXEC
             .
             .
             .
            JSB     EXEC        Transfer control to RTE-III
            DEF     RTN         Return address
            DEF     ICODE       Request code
            DEF     INAME       Name of program to be terminated -- optional
            DEF     INUMB       Type of completion -- optional
            DEF     IPRM1  }
             .              }   Up to five optional parameters -- optional
             .              }
            DEF     IPRM5  }
RTN         return  point       Continue execution (A = as it was, B = as
             .                  it was or parameter address)
             .
             .
ICODE       DEC     6           Request code = 6
            DEC     0           Terminate this program
INAME           or
            ASC     3,name      name = Name of subordinate program to be terminated.
INUMB       DEC     n           n = 0,  Normal completion
                                n = -1, Serial reusability completion. When rescheduled,
                                        program is not reloaded into memory if it is
                                        still resident.
                                n = 1,  Make program dormant but save current suspension
                                        point.
                                n = 2,  Terminates and removes from the time list the named
                                        program. If the program is I/O suspended, the system
                                        waits until the I/O completes before setting the program
                                        dormant; however, this call does not wait. The program's
                                        disc tracks are not released.
                                n = 3,  Terminates immediately the named program, removes it
                                        from the time list, and releases all disc tracks. If
                                        suspended for I/O, the device and channel are cleared by
                                        a CLC. An abort message is printed on the system TTY.
IPRM1  }
IPRM5  }                        Up to five optional parameters to be passed to caller when next
                                scheduled (INAME = 0).
FORTRAN:    DIMENSION INAME(3)  See INAME above
            ICODE = 6
            INUMB = 0           See INUMB above
            INAME(1) = numb B   First two characters
            INAME(2) = numb B   Second two
            INAME(3) = numb B   Last character in upper 8 bits
            REG = EXEC (ICODE,INAME,INUMB,IPRM1. . . IPRM5)
            CALL RMPAR (IPRM1 . . . IPRM5) to pick up the parameters
```

COMMENTS

This call, with its optional parameters, makes it possible for the user to selectively terminate programs he and only he has scheduled. For example, if PROG1 ("Father") schedules PROG2 ("Son") to run, and then later PROG2 schedules PROG3 to run, PROG2 becomes the "Father" to PROG3 (a "Son"). In this case, only the following calls for Program Completion are legal.

- PROG 1 terminates itself or PROG 2
- PROG 2 terminates itself or PROG 3
- PROG 3 terminates itself only.

Option −1 (INUMB = −1) should be used only for programs that have serial reusability. These are disc resident programs that can initialize their own buffers or storage locations. For instance, all library subroutines are serially resuable. When INUMB = −1, the program is reloaded from disc only if it is overlaid by another program. The program must be able to maintain the integrity of its data in memory.

Option 1 (INUMB = 1) is almost the same as a Program Suspend EXEC call. In this case the program restarts from its point of suspension with all resources untouched. Unless the program suspended itself in this way, the program may only be restarted by the program that scheduled it ("Father"), or the ON or RUN operator commands. If the program suspended itself (INAME = 0), it may be restarted by any normal run stimulus (i.e., Schedule, ON, RUN, TIME and Interrupt).

Parameters IPRM1 . . . IPRM5 are optional parameters that are passed to the caller when it is next scheduled. The parameters are passed only when INAME = 0 and may be recovered by a call to RMPAR when the program next executes. In this way a program in the time list may run with the same parameters each time.

Note that the FORTRAN and ALGOL compilers generate a Program Completion EXEC call automatically when they compile 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-III
        DEF     RTN         Return address
        DEF     ICODE       Request code
RTN     return  point       Continue execution (A = as it was, B = as it was or
         .                  parameter address)
         .
         .
ICODE   DEC     7           Request code = 7
```

FORTRAN and ALGOL:

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

COMMENTS

Note that it is illegal to suspend a Batch program with this call (error SC00 results). When a program is suspended (either by this call or the SS command), both the A- and B-Registers are saved and the program is placed in suspension list 6. 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 a GO and parameters, the B-Register contains the address of a five-word parameter array set by the GO request. In a FORTRAN program, a call to the library subroutine RMPAR can load these parameters using the address in the

B-Register as a pointer as long as the RMPAR call occurs immediately after the EXEC call. 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. If you suspect there might not by any parameters, the following example shows how to allow for it.

```
DIMENSION I (5)      Must always specify
                     at least 5
REG = 0.0
REG = EXEC (7)       Suspend
IF (IB) 20,20,10
10 CALL RMPAR (I)    Return Point; get
                     parameters
20 CONTINUE          Return point; no
                     parameters
```

When programming in ALGOL the parameters can be retrieved through RMPAR in the following manner. The variables are declared as integers and then RMPAR is called (immediately after the EXEC call).

```
INTEGER A,B,C,D,E;
    .
    .
    .
CALL EXEC (7)
CALL RMPAR (A)
```

Obtaining the parameters in this manner depends on the compiler placing the contents of A,B,C,D,E in sequential locations.

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

PROGRAM SEGMENT LOAD

Purpose:

To load a background segment of the calling program from the disc into the background overlay area and transfer execution control to the segment's entry point. (See Section IV, Part 8, Real-Time Programming, for information on segmented programs.)

Assembly Language:

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC      Transfer control to RTE-II
                DEF   RTN       Return address
                DEF   ICODE     Request code
                DEF   INAME     Segment name
                DEF   IPRM1   }
                 .            }
                 .            }  Up to five optional parameters
                 .            }
                DEF   IPRM5   }
RTN             return point
                 .
                 .
                 .
ICODE           DEC   8         Request code = 8
INAME           ASC   3,name    name is the segment name
```

Control is transferred to the segment. (A = segment ID segment address, B = as it was or parameter address).

FORTRAN:

```
DIMENSION NAME (3)
ICODE = 8
INAME (1) = numb B      First two characters
INAME (2) = numb B      Second two
INAME (3) = numb B      Last character in upper 8 bits
REG = EXEC (ICODE,INAME,IPRM1 . . . IPRM5)
```

COMMENTS

See Section IV, Overlay Segments and Segmented Programs, for a description of segmented background programs.

On segment entry the registers are set as follows:

A= Segment ID segment address.

B= As it is unless parameters are passed in which case it is the parameter list address (see RMPAR).

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

PROGRAM SCHEDULE

Purpose:

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

Assembly Language:

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC      Transfer control to RTE-II
                DEF   RTN       Return address
                DEF   ICODE     Request code
                DEF   INAME     Name of program to schedule
                DEF   IPRM1  )
                 .           )
                 .           )  Up to five optional parameters
                 .           )
                DEF   IPRM5  )
RTN             return point    Continue execution (A = program status, B = as it
                 .              was or parameter address)
                 .
                 .
ICODE           DEC   numb       9 = immediate schedule, with wait
                                10 = immediate schedule, no wait
                                23 = queue schedule, with wait
                                24 = queue schedule, no wait
INAME           ASC   3,name    name is the name of the program to schedule
IPRM1  )                        Up to five optional parameters
IPRM5  )
```

FORTRAN:

```
                DIMENSION INAME (3)
                ICODE = numb            See ICODE above
                INAME (1) = numb B      First two characters
                INAME (2) = numb B      Second two
                INAME (3) = numb B      Last character in upper 8 bits
                REG = EXEC (ICODE,INAME,IPRM1 . . . IPRM5)
```

COMMENTS

The ICODE parameter determines if the calling program will wait or not, and if the calling program's schedule request will be queued until the scheduled program becomes dormant.

When a program is scheduled, a pointer will be put in its ID segment that will:

a. Point back to the program that scheduled it.
b. Be set to 0 if the program was scheduled by the operator, from an interrupt, or from the time list.

The pointer will be cleared when the program terminates or is aborted. Note that this pointer establishes the program doing the scheduling as the "Father", and the program being scheduled as the "Son".

As soon as a program that had been scheduled with wait completes, the "Father" may recover optional parameter one that indicates if the "Son" was aborted by the system or terminated by the OF operator command. The parameter is set by the system to 100000B and is recovered through RMPAR or a load B Indirect (LDA B,I). However, if the program does not pass back parameters and terminates normally, B will be set as it was on the call.

ICODE = 9 OR 10

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. If the program to be scheduled is a "Son" that was suspended with the EXEC 6 call, some high bits may be set in the A-Register. Only the least 4-bits should be checked for zero in this case.

A schedule with wait (ICODE = 9) call causes RTE-II to put the "Father" in a waiting status (the wait bit is set in the status word in the "Father's" ID segment). If required, the "Father" will be swapped by the system to make way for a program that may run. The "Son" runs at its own priority, which may be greater than, less than, or equal to that of the calling program. Only when the "Son" terminates does RTE-II resume execution of the "Father" at the point immediately following the schedule call.

A disc-resident program may schedule another disc-resident program with waiting, because disc-resident programs are swapped according to their priority when they conflict over use of their core area.

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

A Schedule EXEC Call with no wait (ICODE = 10) causes the specified program to be scheduled for execution according to its priority.

ICODE = 23 or 24

These requests are the same as 9 and 10 except that the system will place the "Father" in a queue if the "Son" is not dormant. When the "Son" becomes available the "Father's" request will be honored. Note that status will not be available in the A-Register and the "Father" will be impeded until the request is honored.

OPTIONAL PARAMETERS

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

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

The Program Schedule EXEC Call is similar to the RU operator request (see Section II). The Execution Time EXEC Call also schedules programs for execution, but without passing parameters.

For the schedule with wait requests (ICODE = 9 or 23), the "Son" may pass back five words to the "Father" by calling the library routine PRTN. For example:

```
PROGRAM SCHEE
DIMENSION IBACK(5)
CALL PRTN (IBACK)
CALL EXEC(6)
```

The EXEC (6) call (which is a termination call) should immediately follow the PRTN call. The "Father" may recover these parameters by calling RMPAR immediately after the "SON" call.

TIME REQUEST

Purpose:

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

Assembly Language:

```
                EXT     EXEC
                 .
                 .
                 .
                JSB     EXEC        Transfer control to RTE-II
                DEF     RTN         Return address
                DEF     ICODE       Request code
                DEF     ITIME       Time value array
                DEF     IYEAR       Optional year parameter
RTN             return  point       Continue execution (A=meaningless, B as it was)
                 .
                 .
                 .
ICODE           DEC     11          Request code = 11
ITIME           BSS     5           Time value array
IYEAR           BSS     1           Year (optional)
```

FORTRAN

```
      DIMENSION ITIME(5),IYEAR(1)
      ICODE = 11
      CALL EXEC (ICODE,ITIME,IYEAR)
```

COMMENTS

When RTE returns, the time value array contains the time on a 24-hour clock, with the year in an optional parameter. The year is a full 4-digit year (e.g., 1974).

Assembler		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).

Another method of obtaining the current time is through a double word load from the system entry point $TIME. $TIME contains the double word integer of the current time of day. If this double word is passed to the library subroutine TMVAL, then TMVAL returns milliseconds, seconds, minutes, and hours. Refer to the Library, Part 7, Section IV.

TIMED EXECUTION (Initial Offset)

Purpose:

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

Assembly Language:

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC      Transfer control to RTE-II
                DEF   RTN       Return address
                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 (negative value)
RTN             return point    Continue execution (A = meaningless, B as it was)
                 .
                 .
                 .
ICODE           DEC   12        Request code = 12
                DEC   0         Put calling program in time list
IPROG               or
                ASC   3,name    name is the program to put in the time list
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) gives the initial offset (negative value)
```

FORTRAN:

```
                DIMENSION IPROG(3)   See IPROG above
                IPROG(1) = numb B    First two characters
                IPROG(2) = numb B    Second two
                IPROG(3) = numb B    Last character in upper 8 bits
                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 (negative value)
                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.

Note that the IOFST parameter is a negative value and must not equal zero.

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.

TIMED EXECUTION (Absolute Start Time)

Purpose:

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

Assembly Language:

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      Transfer control to RTE-II
            DEF   RTN       Return address
            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
RTN         return point    Continue execution (A = meaningless, B as it was)
             .
             .
             .
ICODE       DEC   12        Request code = 12
            DEC   0         Putting calling program in time list
IPROG          or
            ASC   3,name    name is the program to put in the time list
IRESL       DEC   x         Resolution code (1=10's/ms; 2=secs; 3=mins; 4=hrs)
MTPLE       DEC   y         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) = numb B       First two characters
IPROG(2) = numb B       Second two
IPROG(3) = numb B       Last character in upper 8 bits
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 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 }
```

PROGRAM SWAPPING CONTROL

Purpose:

To allow a program to lock itself into core (foreground or background) if the core locks were set up during generation.

Assembly Language:

```
            EXT     EXEC
             .
             .
             .
            JSB     EXEC    Transfer control to RTE-II
            DEF     RTN     Return address
            DEF     ICODE   Request code
            DEF     IOPTN   Control information
RTN         return  point   Continue execution (A = meaningless, B as it was)
             .
             .
             .
ICODE       DEC     22      Request code = 22
IOPTN       DEC     numb    0 = program may be swapped
                            1 = program may not be swapped
                            2 = swap just the program area
                            3 = swap all of the disc resident area
```

FORTRAN:

```
ICODE = 22
IOPTN = numb
CALL EXEC (ICODE,IOPTN)
```

COMMENTS

This call allows the programmer to lock his program into core so it cannot be swapped out for a program of higher priority. Also the programmer can specify if just the program is to be swapped or if the entire foreground/background area is to be swapped with the program.

NOTE

> The program cannot be locked into core if the core lock bits (base page word 1736B, bits 2 and 3) are not set (SC07 error results). The bits are set during generation and can be examined with the SW operator command.

The program's core lock bit (IOPTN = 0 or 1) is set or cleared by this request (refer to ID segment word 15, bit 6 in Table A-2). This bit is also cleared (making the program swappable) if the program aborts or terminates except on the Program Completion EXEC Call where the current suspension point is saved.

The program's core usage bit (IOPTN = 2 or 3) is also set or cleared by this request (refer to ID segment word 15, bit 5 in Table A-2). The bit is initialized when the program is scheduled as follows:

Foreground program — bit is cleared
Background programs — bit is set

The system sets this bit whenever it loads a segment for the program. If the bit is not set, the segment area is not swapped, that is, the segment occupies undeclared core.

When IOPTN = 3, the calling program tells the system that it is going to use undeclared core in its disc resident area. When the program is swapped, the whole disc resident area

is swapped. This allows the program to save working area that it had set up.

When IOPTN = 2, the calling program tells the system that it is not going to use undeclared core in its disc resident area. Only the program itself is swapped.

RESOURCE MANAGEMENT (Resource Numbering)

Purpose:

To allow cooperating programs a method of efficiently utilizing resources through a resource numbering scheme.

Assembly Language:

```
            EXT     RNRQ
             .
             .
             .
            JSB     RNRQ    Transfer control to subroutine
            DEF     RTN     Return address
            DEF     ICODE   Control information
            DEF     IRN     Buffer location
            DEF     ISTAT   Optional parameter
RTN         return  point   Continue execution (A = meaningless, B as it was)
             .
             .
             .
ICODE       OCT     numb    numb is described in Comments.
IRN         BSS     1       Resource number. Returned on allocate; required otherwise.
ISTAT       BSS     1       Status of resources.
```

FORTRAN:

```
ICODE = numb
CALL RNRQ (ICODE,IRN,ISTAT)
```

COMMENTS

Figure 3-6 shows the format of the control word (*numb*) required in the calling sequence.

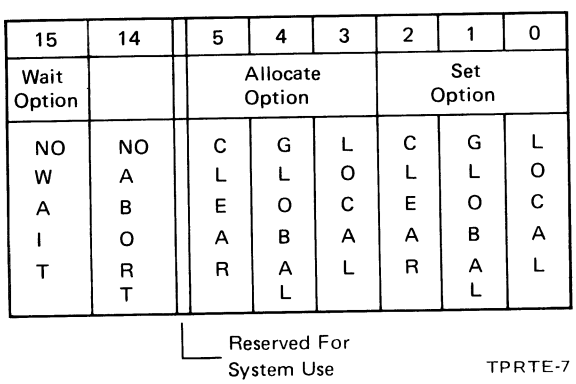


Figure 3-6. Resource Number Control Word Format

If more than one bit is set in the control word, the following order of execution is used:

1. Local allocate (skip 2 if done).
2. Global allocate.
3. Deallocate (exit if done).
4. Local set (skip 5 if done).
5. Global set.
6. Clear.

The status return word (ISTAT) has the following meanings:

ISTAT Value	Meaning
0	Normal deallocate return
1	RN is clear (unlocked).
2	RN is locked locally to caller.
3	RN is locked globbally.
4	No RN available now.

ISTAT Value	Meaning
5	–
6	RN is locked locally to other program.
7	RN was locked globally when request was made.

Note that status 4, 6, and 7 are returned only if the request failed and the "no wait" bit is set.

NO ABORT BIT

The no abort bit is used to alter the error return point of this call as shown in the following example.

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

This special error return is established by setting bit 14 to "1" in the request code word (ICODE). This causes the system to execute the first line of code following the CALL RNRQ if there is an error, or if there is no error, the second line of code following the CALL RNRQ.

ALLOCATE OPTIONS

LOCAL – Allocate an RN to the calling program. The number is returned in the IRN parameter. The number is automatically released on termination of the calling program, and only the calling program can de-allocate the number.

GLOBAL – Allocate an RN globally. The number is released only by a request from any program.

CLEAR – De-allocate the specified number.

The system has a certain quantity of resource numbers (RN's) that are specified during generation. If a number is not available, the program is suspended until one is free, unless the "no wait" bit is set (see the ICODE parameter). If the "no wait" bit is set, the IRN location is set to zero. If the RN allocation is successful, the value returned in IRN is set by the system (it has no meaning to the user) and must be specified (through IRN) when a lock is requested or the RN is cleared or de-allocated.

SET OPTIONS

LOCAL – Lock the specified RN to the calling program. The RN is specified in the IRN parameter. The local lock is automatically released on termination of the calling program, and only the calling program can clear the number.

GLOBAL – Lock the specified RN globally. The RN is specified in the IRN parameter and the calling program can globally lock this number more than once. The number is released by a request from any program.

CLEAR – Release the specified number.

If the RN is already locked, the calling program is suspended (unless the "no wait" bit is set) until the RN is cleared. If more than one program is attempting to lock an RN, the program with the highest priority is given precedence.

If a program makes this call with the "clear" bit set, in addition to either the "global" or "local set" bits, the program will wait (in the general wait list) until the RN is cleared by another program and then continue with the RN clear.

An entry point is provided for drivers or privileged subroutines that wish to clear a global (and only a global) RN.

```
LDA RN
JSB $CGRN
return point
```

LOGICAL UNIT LOCK

Purpose:

To allow a program to exclusively dominate (lock) input/output devices (logical unit, or LU numbers).

Assembly Language:

```
                EXT     LURQ
                 .
                 .
                 .
                JSB     LURQ    Transfer control to subroutine
                DEF     RTN     Return address
                DEF     IOPTN   Control parameter
                DEF     LUARY   LU's to be locked
                DEF     NOLU    Number of LU's to be locked
RTN             return  point   Continue execution (A = lock status, B as it was)
                 .
                 .
                 .
IOPTN           OCT     numb    numb is an octal number:
                                0x0000 – unlock specified LU's
                                1x0000 – unlock all LU's program currently has locked
                                0x0001 – lock with wait specified LU's
                                1x0001 – lock without wait specified LU's
                                x (bit 14) is the no abort bit
LUARY           DEC     xx      LUARY is an array of LU's to be
                DEC     yy      locked/unlocked. Only the least
                DEC     zz      6 bits of each word are used. Ignored when
                 .              IOPTN = 1x0000
                 .
                 .
NOLU            DEC     aa      Number of LU's to be locked/unlocked. Ignored when
                                IOPTN = 1x0000
```

FORTRAN:

```
DIMENSION LUARY (x)
IOPTN = numb
NOLU = aa
CALL LURQ (IOPTN,LUARY,NOLU)
```

COMMENTS

This request allows up to 31 programs to exclusively dominate (lock) an input/output device (e.g., program output to a line-printer). Any other program attempting to use or lock a locked LU will be suspended until the original program unlocks the LU or terminates.

NO ABORT BIT

The no abort bit is used to alter the error return point of this call as shown in the following example.

```
CALL LURQ (IOPTN ... )
GO TO error routine
normal return
```

This special error return is established by setting bit 14 to "1" in the request code word (ICODE). This causes the system to execute the first line of code following the CALL LURQ if there is an error, or if there is no error, the second line of code following the CALL LURQ.

This subroutine calls the Program Management subroutine (RNRQ) for a resource number (RN) allocation. That is,

the system locks an RN number locally to the calling program. Therefore, before the logical unit lock subroutine can be used, a resource number must have been defined during generation. Note that the first 31 Resource Numbers can be used for LU locks.

If the no wait option is coded the A-Register will contain the following information on return.

A = 0 – LU lock successful.
A ≠ 0 – LU lock unsuccessful.
A = –1 – No RN available this time.
A = 1 – One or more of the LU's is already locked.

Note that the calling program may not have LU's locked at the time of this call unless the no wait option is used. Also, all the LU's that the calling program locks are locked to the same RN.

ERROR MESSAGES

When RTE-II discovers an error in an EXEC call, it terminates the program, releases any disc tracks assigned to the program, prints an error message on the operator console, and proceeds to execute the next program in the scheduled list. Table 3-5 is a summary of the possible errors associated with all the EXEC calls. Refer to Appendix E for other system errors.

When RTE-II aborts a program, it prints the following message:

name ABORTED

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

MP *name address*

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

RQ *name address*

An RQ00 error means that the address of a returned parameter is below the memory protect fence.

The following errors have the same format as "MP" and "RQ" errors.

Error	Meaning
TI	Batch program exceeds allowed time.
RE	Re-entrant subroutine attempted recursion.

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 DISC ALLOCATION CALLS

DR01 = Insufficient number of parameters.

DR02 = Number of tracks ≤ zero, illegal logical unit; or number of tracks to release is zero or negative.

DR03 = Attempt to release track assigned to another program.

ERROR CODES FOR SCHEDULE CALLS

SC00 = Batch program attempted to suspend (EXEC (7)).

SC01 = Missing parameter.

SC02 = Illegal parameter.

SC03 = Program cannot be scheduled.

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

SC04 = *name* is not a subordinate (or "son") of the program issuing the completion call.

SC05 = Program given is not defined.

SC06 = No resolution code in Execution Time EXEC Call.

SC07 = Prohibited core lock attempted.

ERROR CODES FOR I/O CALLS

IO00 = Illegal class number.

IO01 = Not enough parameters.

IO02 = Illegal logical unit.

IO03 = Not used.

IO04 = Illegal user buffer.

IO05 = Illegal disc track or sector.

IO06 = Reference to a protected track; or using LG Tracks before assigning Mem (see LG, Section II).

IO07 = Driver has rejected call.

IO08 = Disc transfer longer than track boundary.

IO09 = Overflow of load-and-go area.

IO10 = Class Get and one call already outstanding on class.

ERROR CODES FOR PROGRAM MANAGEMENT

RN00 = No option bits set in call.

RN01 = Not used.

RN02 = Resource number not defined.

RN03 = Unauthorized attempt to clear a LOCAL Resource Number.

ERROR CODES FOR LOGICAL UNIT LOCK CALLS

LU01 = Program has one or more logical units locked and is trying to LOCK another with WAIT.

LU02 = Illegal logical unit reference (greater than maximum number).

LU03 = Not enough parameters furnished in the call.

(This page is intentionally blank.)

Table 3-5. Summary of EXEC Call Errors

ERROR	MEANING	READ 1	WRITE 2	CONTROL 3	PROGRAM TRACK ALLOCATE 4	PROGRAM TRACK RELEASE 5	PROGRAM COMPLETION 6	PROGRAM SUSPEND 7	PROGRAM SEGMENT LOAD 8	PROGRAM SCHEDULE W/WAIT 9	PROGRAM SCHEDULE WO/WAIT 10	TIME REQUEST 11
DR01	Not Enough Parameters 1. Less than 4 parameters. 2. Less than 1 parameter. 3. Number = -1. 4. Less than 3 (not -1).				1	2 4						
DR02	Illegal Track Number or Logical Unit Number. 1. Track number = 0. 2. Logical Unit not 2 or 3. 3. Dealocate 0 or less Tracks.				1	2 3						
DR03	Attempt to release Track assigned to another program.					X						
1000	Illegal Class Number 1. Outside Table. 2. Not allocated. 3. Bad Security Code.											
1001	Not Enough Parameters. 1. Zero parameters. 2. Less than 3 parameters. 3. Less than 5/disc. 4. Less than 2 parameters. 5. Class word missing.	1 2 3	1 2 3	1								
1002	Illegal Logical Unit 1. 0 or maximum. 2. Class request on disc LU. 3. Less than 5 parameters and X-bit set.	1 3	1 3	1								
1004	Illegal User Buffer. 1. Extends beyond FG/BG area. 2. Not enough system memory to buffer the request.	1										
1005	Illegal Disc Track or Sector 1. Track number maximum. 2. Sector number 0 or maximum	1 2	1 2									
1006	Attempted to WRITE to LU2/3 and track not assigned to user or globally, or not to next load-and-go sector. Illegal WRITE to a FMP track.		X									
1007	Driver has rejected request and request is not buffered.	X	X	X								
1008	Disc transfer implies track switch (LU2/3)	X	X									
1009	Overflow of load-and-go area.		X									

PROGRAM SCHEDULE TIME 12	I/O STATUS 13	GLOBAL TRACK ALLOCATE 15	GLOBAL TRACK RELEASE 16	CLASS I/O READ 17	CLASS I/O WRITE 18	CLASS I/O CONTROL 19	CLASS I/O WRITE/READ 20	CLASS I/O GET 21	PROGRAM SWAPPING CONTROL 22	PROGRAM SCHED QUEUE W/WAIT 23	PROGRAM SCHED QUEUE WO/WAIT 24	RNRQ	LURQ
		1	3 4										
		1	2 3										
				1 2 3	1 2 3	1 2 3	1 2 3	1 2 3					
	1 4			1 2 5	1 2 5	1 5	1 2 5						
	1			1 2 3	1 2 3	1 2 3	1 2 3						
				2	2	2	2	1					

Table 3-5. Summary of EXEC Call Errors

ERROR	MEANING	READ	WRITE	CONTROL	PROGRAM TRACK ALLOCATE	PROGRAM TRACK RELEASE	PROGRAM COMPLETION	PROGRAM SUSPEND	PROGRAM SEGMENT LOAD	PROGRAM SCHEDULE W/WAIT	PROGRAM SCHEDULE WO/WAIT	TIME REQUEST
1010	Class GET and one call already outstanding on class.											
LU01	Program has one or more logical units locked and is trying to LOCK another with WAIT.											
LU02	Illegal logical unit reference (greater than maximum number).											
LU03	Not enough parameters furnished in the call. Illegal logical unit reference (less than one). Logical unit not locked to caller.											
RQ00	Return buffer below memory protect fence.	X			X							X
RQ	EXEC call contains an illegal request code. 1. Return address indicates less than one or more than seven parameters. 2. Parameter address indirect through A- or B- Register; 3. Request code not defined or not loaded.	X	X	X	X	X	X	X	X	X	X	X
RN00	No option bits set.											
RN01	Not used											
RN02	Resource number not in Table (undefined).											
RN03	Unauthorized attempt to clear a LOCAL Resource Number.											
SC00	Batch program cannot suspend.							X				
SC01	Missing Parameter. 1. Segment name missing. 2. Not 4 or 7 parameters in Time Call.								1			
SC02	Illegal Parameter 1. Option word is missing or not 0, 1, 2, or 3.						1					
SC03	Program Cannot Be Scheduled. 1. Not a segment 2. Is a segment								1	2	2	
SC04	Attempted to control a program that is not a "son."						X					
SC05	Program Given is Not Defined. 1. No segment. 2. No program 3. "Son" not found.						3		1	2	2	
SC06	Resolution not 1, 2, 3, or 4.											
SC07	Prohibited core lock attempted.											

PROGRAM SCHEDULE TIME 12	I/O STATUS 13	GLOBAL TRACK ALLOCATE 14	GLOBAL TRACK RELEASE 15	CLASS I/O READ 16	CLASS I/O WRITE 17	CLASS I/O CONTROL 18	CLASS I/O WRITE/READ 19	CLASS I/O GET 20	PROGRAM SWAPPING CONTROL 21	PROGRAM SCHED QUEUE W/WAIT 22	PROGRAM SCHED QUEUE WO/WAIT 23	RNRQ	LURQ
								X					
													X
													X
													X
	X	X			X	X	X	X					
X	X	X	X	X	X	X	X	X	X	X	X		
												X	
												X	
												X	
2													
									1				
										2	2		
2										2	2		
									X				

SECTION IV
REAL-TIME PROGRAM PREPARATION

INTRODUCTION

This section is divided into seven parts that describe the operating procedures and formatting conventions of background programming aids of the Real-Time Software. The memory requirements stated apply to the program plus a reasonable area for symbol tables where appropriate.

PART 1. RTE EDITOR

The RTE-II Editor creates, lists, and edits symbolic source language tapes and disc files.

PART 2. RTE INTERACTIVE EDITOR

The RTE-II Interactive Editor is a foreground/background swappable program that creates, lists, and edits symbolic source language tapes and disc files. The editor requires the RTE File Management Package option.

PART 3. RTE FORTRAN

The FORTRAN compilers accept source programs from either an input device or a source file created by the RTE Editors and translates the source programs into relocatable object programs. The relocatable code is punched on paper tape or stored in the LG tracks of the disc or both.

PART 4. RTE ALGOL

The ALGOL compiler accepts source programs from either an input device or a source file created by the RTE Editor, and translates the source programs into relocatable object programs. The relocatable code is punched on paper tape or stored in the LG tracks of the disc or both.

PART 5. RTE ASSEMBLER

The Assembler accepts source programs from either an input device or a source file created by the RTE Editors, and translates the source programs into either absolute or relocatable object programs. Absolute code is punched in binary, suitable for execution outside of RTE-II. The relocatable code is punched on paper tape or stored in the LG tracks of the disc or both.

PART 6. RTE-II LOADER

The loader accepts relocatable object programs from either an input device, or a file created by the Assembler, ALGOL, or FORTRAN compilers on LG tracks. The program can optionally be loaded into the background and run; or the program can be loaded into the background with the DEBUG library routine linked to it; or the program can be loaded into the disc-resident user program area.

LG TRACK AREA

The loader also provides the facility for compilation or assembly, loading, and executing a user program without intervening object paper tapes. To accomplish this, the compiler or assembler stores the relocatable object code, which it generates from source statements, on the disc in a predefined group of tracks called LG Tracks (see LG operator request). Then separate operator requests initiate loading (RU,LOADR) and execution (RU,program). All of the operating procedures have optional parameters that specify source input, relocatable output, and list device. These parameters take the form of logical unit numbers as follows:

Logical Unit Number	Function
0	Bit Bucket
1	System Teleprinter
2	System Disc
3	Auxiliary Disc

4	Standard Punch Device
5	Standard Input Device
6	Standard List Device
7	Can be assigned to any devices by the user, for the defined range of logical units.
8	
9	
10	
.	
.	
.	
63_{10}	

Note that LU8 is recommended as the magnetic tape device.

PART 7. RTE RELOCATABLE LIBRARY

This part describes the libraries used by RTE-II, re-entrant subroutine structure, privileged subroutine structure, and utility subroutine structure.

PART 8. SEGMENTED PROGRAMS

This part describes the procedures for writing segmented programs in Assembler, ALGOL, and FORTRAN.

PART 9. MULTIPLE TERMINAL OPERATION

This part describes the operation and configuration of the multi-terminal moniter.

PART 1
RTE-II Editor

INTRODUCTION

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

EDITOR INPUT/OUTPUT FILES

In general, the Editor requires two inputs: an edit file containing edit commands, and a symbolic file containing source programs, on tape or disc. Both the edit file and the symbolic file may consist of more than one physical tape. The RTE-II Editor processes all properly related inputs and produces a single updated file on a paper tape, or a disc file, or a list of the source file. Both the symbolic file and the updated file consist of a series of records; each record contains 1 to 72 ASCII characters. In a disc file, records are packed within a track, and tracks are chained together.

EDITING PROCESS

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

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

OPERATING PROCEDURES

The choice of input device for the edit file depends on the complexity of the editing. For only one or two short edit commands, using a teleprinter keyboard is faster; but for a longer edit file, it is faster to punch the file on tape and read it through the photo-reader. If the photo-reader is used, the edit file tape must be in place before scheduling the editor with an RU operator request.

RU, EDIT

Purpose:

To schedule the editor for operation.

Format:

RU, EDIT, *input 1, input 2, output, edit*

Where:

input 1 Corrections. The logical unit number of the edit file input device. The standard input (logical unit 5) is used if none is specified (must not = 2). If the edit file is entered through the system teleprinter, the following message will be printed:

/EDIT:ENTER EDIT FILE:

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

output The logical unit number of an output device for the updated file or the listing (see edit). For a disc file, *output* = 2. If defaulted, logical unit 6 is used.

edit The type of Editor operation:

If *edit* = 0 (or no value), a normal edit.
If *edit* = 1 the symbolic file is only listed (output may not be 2).
If *edit* = 2 a disc file is generated from the symbolic file (output is ignored). If *input 2* = 2, the old file is not released.

Example:

RU,EDIT<is equivalent to RU,EDIT, 5,5,4,0>

MESSAGES TO OPERATOR

At the end of the edit file, the Editor prints the following message and suspends itself.

/EDIT: END EDIT FILE

The sysmbolic file must be loaded into the correct device to resume operations. If the updated file is paper tape, the Editor generates blank leader on the output tape. Enter:

GO, EDIT

When a paper tape is being input through the tape reader, RTE Driver DVR00 can interpret an end-of-tape (EOT) in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how DVR00 subchannels were configured during generation. In any case, an EOT suspends the editor. For more information refer to the DVR00 manual (HP Part No. 29029-95001). To change DVR00 Subchannel assignments, use the LU command.

If an EOT causes the tape reader to be set down, the RTE system will output a message to the operator:

I/O ERR ET EQT *#eqt*

The operator must up the tape reader with the UP operator command.

UP, *eqt*

If an EOT does not cause the tape reader to be set down, the system does not output any message. However, the editor is still suspended. Editing can continue or terminate at this point by the operator entering:

GO, EDIT, *numb*

Where:

numb if 0 (or not given), means to read the next tape; if *numb* is any other number, it means terminate the editing process.

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

/EDIT: TRACKS IN NEW FILE
/EDIT: *disclu, trk numb*

Editing finally terminates with the messages:

/EDIT: END OF EDIT RUN

Where

disc lu is the disc logical unit number and *trk numb* is the track number. This formation is used with the LS operator request.

The Editor must be rescheduled with the RU command for the next operation.

EDIT COMMANDS

The edit commands, consisting of ASCII characters terminated by a RETURN (and if paper tape input a LINE FEED), direct the editing process and have this format:

/ ee, p1, p2, p3

Where

The first non-blank character must be a slash (/).

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

p1 through *p3* are either record sequence numbers or character numbers referring to the symbolic file.

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

EDIT COMMAND FORMATS

Format	Function
/I,*r*	Insert new records after record *r*. The new records follow the /I command in the edit file.
/D,*r1* [,*r2*]	Delete record *r1*, or records *r1* through *r2*, inclusive.
/R,*r1* [,*r2*]	Replace record *r1*, or records *r1* through *r2*, inclusive. The replacement records follow the /R command.

Format	Function
/CI,*r*,*c*	Insert new character(s) after character *c* in record *r*. The new characters follow the /CI command.
/CD,*r*,*c1*,[,*c2*]	Delete character *c1*, or characters *c1* to *c2*, inclusive, in record *r*.
/CR,*r*,*c1*,[*c2*]	Replace character *c1* to *c2*, inclusive, in record *r*. With the character(s) following the /CR.
/E	Ends the edit file, or, without other commands, dumps a file from the disc on the output device or copies a tape.
/A	Aborts editing – useful only when entering the edit file from teleprinter.

EDIT FILE EDITING

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

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

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

tells the Editor to "replace record 6 (in the first edit file) with the record "/D, 314, 315."

EDITOR ERROR MESSAGES

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

/EDIT: error message: illegal edit command

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

Error Message	Meaning
MEM OVERFLOW	The edit file overflows available memory; the Editor prints the command causing the overflow Edit terminates.
CS ERR	Illegal edit command, which is printed.
PARAM ERR	Edit command "r" or "c" is illegal: non-numeric, = 0,> 72, $r_2 \leq r_1$, $c_2 \leq c_1$; command printed.
SEQ ERR	"r" parameter ≤ a previous "r" or "r" greater than range of symbol file; command printed.
/I ERR	No insert source statements after /I; command printed.
/R ERR	No replacement statements after /R; command printed.
/C OVF	Character overflow in edit statement (i.e.,> 72 characters).
DISK OVF	No disc space for file; edit terminates.
FILE UN	Undefined symbolic file, or LUN (edit file) = 2. Edit terminates.

PART 2
RTE Interactive Editor

INTRODUCTION

The RTE Interactive Editor is a foreground/background swappable program that uses all of foreground/background memory (exclusive of itself and its subroutines) as disc I/O buffers. The Interactive Editor requires at least 4.5K of background or foreground memory and the RTE-II Batch Spool Monitor Option in order to operate properly.

SOURCE/EDITED FILES

The Interactive Editor obtains the source file to be edited from the user, places the source file into the working area, and then turns control over to the user for the editing process. Note that when the editor is turned on it releases all the tracks it owns if and only if it does not own the current LS tracks. The source file can be obtained from:

- The system LS tracks. If the LS tracks are empty (or released by the LS,0 operator command) the source file may be input directly through the keyboard.
- A source file (type 3 or 4) created by the File Manager. Note that the file can also be a type 0 file (e.g., the tape reader).

As the user edits the source file the results are placed into the destination or edited file. At the end of the editing process the user can place the destination file into the LS area for use by compilers and the File Manager and/or create or replace a file "name" within the File Manager area.

The editor performs edits by reading lines from the source file (which has been moved to a work area on the disc), and modifying them under control of the editor commands. This results in a destination file which is located in the memory buffer. When the buffer becomes full it is transfered to the disc work area. With the exception of the "J" command it is not possible to back up in the source file; however, several commands result in the editor copying some, none, or all lines to the destination file, abandoning the original source file, and making the destination file the source file from which to create a new destination file. In this manner many passes can be made with the editor and one can effectively back up. For convenience, the act of passing all remaining lines to the destination file and then abandoning the source and making the destination the new source will be called rolling over.

DESCRIPTION

The Interactive Editor is a powerful tool provided to the user as an aid in the process of editing programs. The power comes from a great many commands designed to manipulate the source file before the actual edit takes place. Many of the commands have default values that can be disasterous if the user is not aware of them. One example is the "D" command. If the user does not correctly specify the optional parameters that go with the "D" command, part or all of the source file can be eliminated.

For this reason the user should always maintain a backup of the current source file that is being edited. If the editing process is lengthy, it is recommended that intermediate results be stored in a file for backup should the current source be eliminated. This prevents having to start from scratch should something happen.

A full understanding of the Interactive Editor and its many capabilities will only be possible through actual use by trial and error. For this reason it is strongly recommended that the user create some scratch files and experiment with all of the editor's commands.

Before beginning the edit process, it is recommended that the user obtain a listing of the source file complete with line numbers. This can be accomplished with the File Manager LI Operator command.

RU,EDITR

Purpose:

To schedule the Interactive Editor for operation.

Format:

RU,EDITR[,*ttylu*[,*line length*]]

Where:

ttylu = The logical unit number of the teletype device to be used for command input. Default is logical unit 1.

line length = Maximum output record length in characters. Default is 150 characters.

MESSAGES TO OPERATOR

As soon as the editor is scheduled it requests information on the source file.

SOURCE FILE?
/*file name* [:*security*[:*cr numb*]]
– or –
Λ

Where

/ is a prompt character supplied by the editor.

file name is the name of a File Manager file. The file name can be a type 0 file.

Λ is a symbol representing a blank (space bar). If a blank is entered for *file name* the input comes from the LS tracks.

cr numb is the cartridge reference (CR) number that is a numeric identifier assigned to all cartridges in the system.

When the editor requests information on the source file, the only two legal responses are a blank or an existing file name. Any other response will cause a FMGR –006 error (file not found). If the user desires to abort the editor at this time enter a blank, and then enter an A in response to the next prompt (/).

If a legal file name is used for the source file question the editor obtains the file from the File Manager, puts it in the working tracks and prints the first line of the file on the TTY. This is the pending line and editing can now begin. Note that if a security code and cartridge number are specified as part of the file name, the editor retains them and uses them as the default values when the edit is completed with the EC*name* command. If new security code and cartridge number values are specified in the EC*name* command, they override the original values.

If a blank is used for the source file question the editor obtains the file from the LS tracks, puts it in the working tracks and prints the first line of the file on the TTY. This is the pending line and editing can now begin. If the LS pointer has not been set, or is pointing to a track that contain an EOF as the first record of a file, the editor prints EOF, which indicates end-of-file. If the LS tracks have not been previously used for logical source, they will not be properly formatted and the editor will abort with the message CORRUPT FILE. In this case the tracks can be formatted by the following command.

*LS,0

The above command formats the empty LS tracks so that when the editor is scheduled, and the LS tracks are referenced as input, the user can enter a file on-line through the keyboard. The file is entered after the EOF message. Enter a space following the slash prompt and begin typing the text of the file. See the example below.

`*ON,EDITR`	Schedule the editor
`SOURCE FILE?`	Location of source file
`/Λ`	Use LS tracks
`EOF`	LS not assigned
`/ΛASMB,L,T`	Λ = insert after pending line
`/Λ;NAM PROG1`	; = first tab, 7th column and next
`/Λ;EXT EXEC;xxx`	; = second tab, 21st column. xxx stands for comments.

EDITOR COMMANDS

Table 4-1 is a summary of the Interactive Editor commands for quick reference. Each of the commands shown in the table are described in more detail, some with examples, in the following paragraphs.

Table 4-1. Summary of EDITR Commands

Control Commands	Description
/	Prompt character printed by the Editor
CTRL G	Invoke or delete bell ring
M*name*	Merge source file called *name* after pending line.
T (;)	Set tab stops
W	Set window (column) boundaries
#	Add line sequence numbers
=*n*	Set line length to *n*.
K	Delete trailing blanks
Search Commands	**Description**
CTRL @	Find field in a zero length line
ESC	Find field of indefinite length
B<*find field*>	Find *find field* thru beginning of file-saves
D<*find field*>	Delete lines until *find field* or EOF
F<*find field*>	From pending line find *find field* or EOF
J<*find field*>	Find *fine field* thru beginning of file-deletes
Exchange Commands	**Description**
G	Character replace on pending line
Y	Exchange pending line, display next occurance of line

Table 4-1. Summary of EDITR Commands (Cont.)

X	Enable exchange pattern all lines (list)
Z	Enable exchange pattern all lines (no list)
V	Unconditional character replace (list)
U	Unconditional character replace (no list)
Character Edits	**Description**
C<*text*>	Edit pending line and go to next line
P<*text*>	Edit pending line and leave as pending line
CTRL R	Replace characters
CTRL I	Insert characters
CTRL S	Insert characters (alternate for CTRL I)
CTRL C	Delete characters
CTRL T	Truncate line
Relative Edits	**Description**
+*n*	Space down *n* lines
/*n*	Space down *n* lines
–*n*	Delete *n* lines
Pending Line Edits	**Description**
R<*text*>	Replace pending line with *text*
I<*text*>	Insert *text* before pending line
A<*text*>	Insert *text* after pending line
O (oh)	Duplicate pending line and edit
N	Print pending line number

Table 4-1. Summary of EDITR Commands (Cont.)

List Commands	Description
P	Display pending line
L*n*	List *n* lines
+	List the next line
/	List the next line
n	Go to line *n*
S	Print the approximate number of words in the destination file (to pending line −1).
↑*n*	Go back *n* lines in the destination file.
ND	Print current line number in the destination file.
H	Print the number of characters in the pending line.
Terminate Commands	**Description**
A	Abort EDITR
EL	Edited file placed in LS tracks
EC*name*	Create *name* and store edited file there
ER	Replace old file with new file (retaining same name)
ER*name*	Replace file *name* with edited file
ELR	Replace old file with new file (retaining same name) and also store in LS tracks
ELC*name*	Create *name*, store edited file there and in LS tracks
ELR*name*	Replace file *name* with edited file and also store in LS tracks

CONTROL COMMANDS

PROMPT CHARACTER

The editor prompts with a slash character "/" and bell. The prompt is also used as a delimiter in some of the commands. That is, if you are changing some characters in a line and you want to save some as they are, you use the prompt. Both the prompt and bell can be changed. The prompt character can be changed with the exchange command (X) and the bell is turned on or off with the CTRL G command. Note that while the prompt can be changed, the slash (/) is exclusively used as a command to space down and is not changed. For example, if you change the slash prompt to a $ prompt with the "X" command, entering a slash will still space down *n* lines.

CTRL G (CONTROL G)

When the editor is scheduled the bell is rung with every prompt. To turn the bell off (or turn it back on), enter CTRL G.

M*name* (MERGE SOURCE)

The entire contents of the file called *name* is inserted after the pending line and before the next line. The next line then becomes the pending line. If the LS area is specified an error indication is given. If the file is not found, nothing is inserted and FMGR-6 error is given. The next line still becomes the pending line.

; (TAB STOPS)

The initial tab character when the editor is turned on is the semicolon (;) (not appropriate for editing ALGOL). With the stop set for the 7th and 21st columns. The tab character is changed with the T command as follows:

/T*x*[*s1*],*s2*[,*s3*. . . *s10*]]. . .]

Where

x is the new tab stop control character (replaces original semicolon).

s1–*s10* are the column numbers of the stops.

Tabs beyond the highest defined stop are replaced with blanks. The tab character may be changed without changing the stops. For example:

/T %

changes tab character to a percent sign (%) without changing the stops.

W (WINDOW)

The initial window field when the editor is turned on is columns 1 through 150. The field is used to limit the character exchange, and one form of the find field, to patterns found beginning in specific columns. For example:

/W7,9

Where

7 is the starting column of the window.

9 is the last column of the window.

With the window set thus, the Search command "F" can be used to find a line with a specified pattern. For example, you want to find the first line from your present position that contains a NAM statement starting in column 7, 8, or 9.

/W7,9
/F/NAM

#(SEQUENCE NUMBERS)

This command adds sequence numbers to all lines in the file. The three character identifier starts in column 73 and the numbers start in column 76. Note that the TTY has a line length limit of 72 characters which must be taken into consideration. However, when using a CRT terminal it automatically does a line feed and places the sequence numbers on the second line or in between each line in the file. The command syntax is as follows:

/#[*xxx*] [*numb1* [,*numb2*]]

Where

xxx is the three character identifier. It occupies columns 73-75 and must be accounted for when specifying *numb1* and *numb2*. If the entire command is allowed to default (e.g., /#), the numbers will start in column 76.

numb1 is the starting number. The first line will start with this number. If allowed to default it starts at 00000.

numb2 is the incrementing value. *numb1* is incremented by *numb2* for each line. If allowed to default, *numb1* is incremented by 10.

An example of using the sequence number command is:

/#ΛΛΛΛ01

There is no three character identifier (Λ is a space). The line numbers start at 00000 (4th Λ = default) and are incremented by 1. Note that the "#" command always rolls over and always ends with an EOF message.

=*n* (SET LINE LENGTH)

The line length is initially set to 150 characters (see *line length* in the ON,EDITR command) when the editor is turned on. This command resets the line length to the value *n*.

K (KILL TRAILING BLANKS)

This command deletes all trailing blanks in the file. A totally blank line will be reduced to two blanks. Note that the "K" command always rolls over and always ends with an EOF message.

SEARCH COMMANDS

FIND FIELD

The find field consists of *n* characters (where *n* is greater than or equal to 0) typed after a "B," "D," "F," or "J" command. The editor will search for a line containing a matching field in the specified position(s). Only as many characters as specified are used in the match. Options used in specifying the find or matching field are:

NULL – If no field is given the last find field entered is used.

ESC – The escape character is a find field of indefinite length and eliminates the need for specifying a specific find field. For example:

/FESC XXX ESC YYY

The Escape character tells the editor to find a line contaning "XXX" followed by "YYY." "XXX" may be anywhere in the line and "YYY" is constrained only to follow the "XXX" but otherwise may be anywhere in the line. Note that on some terminals the Escape Key generates an action within the terminal that is not desirable. In this case, use the alternate Escape Key, ~ (sine wave).

/ – Delimiter character (/ is default) used as the first character of the find field acts the same as an ESC character but the search for the match is limited to patterns beginning within the window established by the "W" command.

CTRL@ – Control @ as the find field will find a zero length line.

TAB – The find field is tabbed according to the established TAB STOPS. Tabbed over characters are replaced by blanks in the find field.
For example:

```
LABEL   LDA B,I
        LDA B,I
```

are instructions located in the source file. The following search command,

/F;LDA

will find all instructions LDA but not LABEL LDA (or any other LDA with a label in front of it).

B<FIND FIELD>(SEARCH FROM BEGINNING)

Search from the beginning of the file and find the first line which matches the *find field* and make it the pending line. All lines passed over are put in the destination file (i.e., lines are not deleted with this command). If the find field is not found the search ends at EOF. The "B" command rolls over the file, disables any exchange option set up, and then searches for the *find field*.

D<FIND FIELD>(DELETE)

Delete the pending line and all lines down to the line containing the *find field*. The line containing the *find field* becomes the new pending line. If the *find field* is not encountered the remainder of the file is deleted. If the *find field* is null then the last *find field* entered is used.

F<FIND FIELD>(FIND)

Search from the pending line to the end-of-file for the line containing the *find field* and make it the new pending line. If the *find field* is not encountered the search ends at the EOF, lines are not deleted with this command.

J<FIND FIELD>(JUMP)

Save the pending line and then jump to the first line containing the *find field* and make that line the pending line. The "J" command may be used to either delete or copy lines in the file. If the jump terminates after the current pending line, lines are deleted; if it terminates prior to the pending line then the lines between the jump target and the pending line when the jump was given will be passed over again; thus a copy is effected.

EXCHANGE COMMANDS

The "G," "Y," "X," and "Z" commands set up two character strings. Wherever the first string is encountered it is replaced by the second string. Exchanges are made only if the first character of the first string is within the window established by the "W" command (i.e., the string must start in the window).

The "U" and "V" commands set up an exchange field length and a character string. The characters in the exchange field are replaced by the character string. The exchange field starts at the first position of the window but is not limited by the length of it.

For the "X," "Z," "U," and "V" commands, the exchange does not take place until another command is entered after the exchange command (e.g., a search command). The number of lines in which the exchange occurs depends on this command. In general, an exchange occurs on all lines passed over during a search or positioning to a new pending line. The exception is when the "B" command is used and causes the file to roll over. This disables any exchange option set up, and then searches for the *find field*. The exchange takes place from the pending line to the EOF but not after the roll over. An exchange will not occur on lines inserted by the "I" command just prior to an exchange command. Inserted lines may be included in the exchange by repositioning the pending line to a line above the inserted lines before entering an exchange command.

The "G" and "Y" commands are the same as the "X" command but do an exchange on the current pending line only and are executed immediately.

An exchange pattern remains in effect until a new one is entered.

The editor generates records in word-length (two characters per word). For example, a source file which contains the record "ERR 1" will be changed on a pass through the editor to "ERR 1Λ." This addition of a trailing blank usually has no significance except during the exchange process. For example, /XERR 1/ would set up an exchange to delete all records ERR 1, but would leave the trailing blank.

This trailing blank is then padded to two blanks by the editor during the exchange.

The exchange fields are not tabbed, thus the tab character is equivalent to any other character in an exchange string. Also the first delimiter delimits the two fields, subsequent delimiters are treated as normal characters of the "new" field.

G<OLD FIELD>/<NEW FIELD>

The "G" command is an action command and performs an immediate exchange (substitutes *new* for *old*) on the pending line and leaves it as the pending line. The first string may be any length except zero, and the second string any length including zero.

Y<OLD FIELD>/<NEW FIELD>

The "Y" command causes an exchange of *new* data for *old* data in the pending line only. The first string may be any length except zero, and the second string any length including zero. The editor then finds the next occurence of the *old* data and prints that as the pending line. To edit that line enter the "Y" command alone; the data is not required. To skip that line and find the next occurence of the *old* data enter the "F" command alone; the data is not required.

X<OLD FIELD>/<NEW FIELD>

The "X" command enables an exchange of *new* data for *old* data. The first string may be any length except zero, and the second string any length including zero. The exchange takes place when the next command is entered, and all lines where an exchange takes place are printed. The "X" command will also change the prompt or exchange character (which is a slash (/) when the Editor is turned on). For example, /X$ will change the prompt/exchange character from a slash (/) to the dollar sign ($).

Z<OLD FIELD>/<NEW FIELD>

The "Z" command will enable an exchange of *new* data for *old* data. The first string may be any length except zero, and the second string any length including zero. The exchange takes place when the next command is entered, and all lines where an exchange took place are not printed. This command is the same as the "X" command except it does not print lines in which the exchange has been made.

U<*char1 char2 . . . charn*>/<NEW>

Sets up a replace of any number of characters including zero (specified by *char1 char2 . . . charn*) beginning at the starting column of the window. The string *char1 char2 . . . charn* is not used as a pattern for a search but is used only to specify the number of characters to be replaced in the window by the *new* pattern. Lines in which the exchange is made will not be listed. For example, to change the first character of every line to the File Manager command ":LI," enter:

```
/W1          Set window to 1st column

/U1/:LI,     Replace 1st character with :LI,

/FEND        Do every line until END found
```

This command does unconditional:

a. inserts (first field zero length).
b. deletes (second field zero length).
c. exchanges.

V<*char1char2 . . . charn*>/<NEW FIELD>

This Exchange command is the same as the "U" command except that all lines in which the exchange is made will be listed.

CHARACTER EDITS

There are two types of Character Edits, the "C" command and "P" command. Within these two types are four modes of Character Edits, replace (CTRL R), insert (CTRL I), delete (CTRL C) and truncate (CTRL T). The initial starting mode is replace. The mode may be changed at any time in the line by entering another mode command. Note that the Mode Control characters are non-printing (i.e., CTRL I does not cause a character to be printed on the teletype).

C<TEXT>

The "C" command will edit the pending line, display the results of the edit, pass the edited line to the destination file, then display the next line of code as the pending line.

P<TEXT>

The "P" command will edit the pending line, display the results of the edit, and leave the altered line as the pending line.

CTRL R (REPLACE)

CTRL R stands for Replace Mode and is the default mode. In the Replace Mode each character is replaced by the new character. Characters may be skipped by entering the prompt for each character to be skipped. Using the Tab

character will also cause characters to be skipped. Note that skipped characters appear in the new line the same as they did in the old line. For example:

```
      NAM PROG 1     Old line of code

/P;/////////2  Tab to 7th column (skip) then skip the next
               9 columns and change the 1 to a 2.

      NAM PROG 2     Result is displayed.
```

Note that in the above example CTRL R was not entered – it is the default mode.

CTRL I (INSERT)
–or–
CTRL S (INSERT)

CTRL I or S stands for Insert Mode. CTRL S is used on terminals where CTRL I has a special function (e.g., the HP 2754 (ASR35) teleprinter uses CTRL I as a tab function). In the Insert Mode each new character is inserted in the line immediately before the character under which the CTRL I or S is entered. Once the CTRL I or S is entered character skipping will insert blanks. The Tab character will also insert the tabbed number of blanks. For example:

```
          NOP  Old line of code

/P①Label       The word LABEL is inserted in the line
               starting in column 1.
```

Note that CTRL I or S is a non-printing character and appears in the example with a circle around it for clarity. Since characters cannot be skipped in the Insert Mode all skipping must be done first (in the default Replace Mode) before the mode is changed to Insert. For example, it is desired to insert a letter in a word (or a word in text).

```
BUFER   DEC 6
        DEF BUFR    Old lines of code.

/P/////////////①E  Insert "E."

        DEF BUFER   New line of code.
```

CTRL C (DELETE)

CTRL C stands for Delete Mode. In the Delete Mode each character, or place holder, entered following the CTRL C will delete a character in the pending line. The Tab character will delete everything up to the tab stop (i.e., character skipping will delete characters). When Carriage Return is entered the pending line will be left justified. For example, to change a spelling error in the comments of a listing.

```
                SSA,RSS   SSKIP IF NEG
/P;;©X
                SSA,RSS   SKIP IF NEG
```

CTRL T (TRUNCATE)

CTRL T stands for truncate the line. When this mode control character is entered the remainder of the line will be eliminated.

RELATIVE EDITS

+n(,lu) or */n(,lu)*

The "+" or "/" command causes the *n*th line following the pending line to be displayed and made the pending line. If *n* is missing, default is the next line following the pending line. If *lu* is given, any exchange prints are done on *lu*; otherwise they are printed on the command device.

-n

The "-" command deletes *n* lines. If *n* is missing one line is deleted.

PENDING LINE EDITS

R<TEXT>(REPLACE)

The "R" command will replace the pending line with *text*. If no *text* is given the new line has zero length.

I<TEXT>(INSERT BEFORE)

The "I" command will insert a new line of *text* in front of the pending line. If no *text* is given the new line has zero length.

Λ<TEXT>(INSERT AFTER)

The "Λ" (space bar) will insert the new line of *text* immediately after the pending line. If no *text* is given the new line has zero length.

O<TEXT>(COPY AND EDIT)

The "O" command places the pending line in the destination file, then performs a /P <TEXT> on a copy of that pending line. The result (in the destination file) is two copies of the pending line, one unchanged and one edited with *text*. For example:

BUFR1 DEC 50	Current pending line.
/OBUFR2	Save pending line and perform P <text> edit.
BUFR2 DEC 50	Result on teletype.

In the destination file the results would be

BUFR1 DEC 50
BUFR2 DEC 50

N(LINE NUMBER)

The "N" command causes the current pending line number to be printed. This number is based on the first line of the file being line number one, the second line number two and so on. These are the line numbers shown on a listing of the file obtained with the File Manager LI command. These line numbers are not the same as those given with the "#" command previously described unless the format /#*xxx*1,1 is used. These numbers are restarted each roll over; thus any inserts or deletions will change the numbers.

LIST COMMANDS

P(DISPLAY PENDING LINE)

The "P" command entered by itself causes the pending line to be printed on the system TTY.

L*n*(,*lu*)

This command will list *n* number of lines on logical unit *lu*. The listing device must be an output type device (e.g., TTY, line printer, tape punch, etc.) If *lu* is not a legal device, the editor will continue to run until it has passed *n* lines. Note that each line has two blanks as the first characters.

/(SLASH)

Entering the "/" or "+" character alone will cause the next line following the pending line to be displayed. Entering the "/" or "+" character and a number will cause that number of lines to be skipped (i.e.,/*n* is the same as +*n*).

n(GO TO *n*)

In this command only the line number need be entered. The line number entered is displayed and becomes the pending line. These line numbers are the same as those described under the "N" command. If *n* is less than or equal to the number of the pending lines a roll over is performed. Zero is interpreted as one.

H(CHARACTER COUNT)

The "H" command prints the number of characters in the pending line. This number is always even for a "new" pending line, but increments/decrements as characters are added/deleted. Since operations such as Gxxx/AAA remove xxx and insert 3 blanks at the end of the line, the "H" command immediately following the "G" command will show a decrement of 3 characters. But if the file is rolled over the same line examined again, it will be the original length until a "K" command deletes the trailing blanks or the file is recycled with the File Manager.

S(WORD COUNT)

The "S" command prints the approximate number of words in the destination file (to pending line −1). An example use of this command could be to break up large paper tapes. A paper tape can hold a maximum of approximately 14,000 words before exceeding the box size. After determining the word count with the "S" command the tape can be broken with a series of 0-length records.

↑*n*(BACK UP)

The "up arrow" command backs up in the destination file. "Up arrow" by itself backs up 1 line. If *n* is too large (i.e., back up beyond the top of the file), the error message ?? is displayed. Note that the "up arrow" command forces a transfer from the source file to the destination file. This could cause a loss of data if used in conjunction with the "J" command. Refer to the "J" command for more information.

ND(PRINT CURRENT DESTINATION LINE NUMBER)

The "ND" command prints the current line number in the destination file. By implication, N by itself prints the pending line number in the source file.

TERMINATE COMMANDS

The Editor terminate commands assign the edited file to the RTE system Logical Source LS tracks, a file name in the File Manager, or both. Note that once the character "E" is entered the editor begins its termination process. No other command (except rubout or "A") may be entered at this time except one of the following legal termination commands.

The only legal command for dumping the edited file to a type 0 file is the ER or ELR command.

A(ABORT)

The editor is aborted and the original source file remains unchanged.

EL

This command ends the edit and assigns the edited file to the LS tracks. The location of the LS tracks is returned to the user, and also set up on the base page.

EC*name*(:*security*(:*cr numb*))

This command ends the edit, creates a file called *name* and stores the edited file in *name.* LS tracks are not used. Note that if a security code and cartridge number are specified as part of the file name in response to SOURCE FILE? when the editor was turned on, these numbers will be used to create the new file name if ECname is entered by itself. If this is not desired, then use ECname (:security (:cr numb))

Where:

security = 0 or a number

cr numb = 0 or a number

ER
–or–
ER*name*(:*security*(:*cr numb*))

This command ends the edit and, if *name* is not supplied, assigns the edited file to the original file name. If *name* is supplied the edited file is assigned to the file called *name.* A file is not created with this command, but replaced (i.e., old file contents are purged and new edited file contents take their place). Note that if a security code was used in the old file's name it must be supplied, otherwise an FMGR-7 error occurs when EDITR attempts to write new contents into the old file.

ELC
–or–
ELC *name (:security(:cr numb))*

This command ends the edit, creates a file called *name*, and stores the edited file in *name* and the system LS tracks. The location of the LS tracks is returned to the user. Note that if a security code and cartridge number are specified as part of the file name in response to SOURCE FILE? when the editor was turned on, these numbers will be used to create the new file name if EC *name* is entered by itself. If this is is not desired, then use EC*name (:security(:cr numb)).*
Where:

security = 0 or a number

cr numb = 0 or a number

ELR
–or–
ELR*name*(:*security*(:*cr numb*))

This command ends the edit and, if *name* is not supplied,assigns the edited file to the original file name and the system LS tracks. If *name* is supplied the edited file is assigned to the file called *name* and the system LS tracks. A file is not created with this command, but replaced (i.e., old file contents are purged and new edited file contents take their place). Note that if a security code was used in the old files name it must be supplied, otherwise an FMGR-7 error occurs when EDITR attempts to write new contents into the old file.

EDITR ERROR MESSAGES

Error Messages	Meaning
??	Error in command just given EDITR, or input device has timed out waiting for a command.
EOF	A command has caused an attempt to read beyond the current end of the source file.
CORRUPT FILE	Input record length from the system LS area is greater than 150 characters, or was not properly formatted.
FMGRxxx	An error detected by the File Manager routines. Refer to the Real-Time Batch/Spool Monitor Manual.

PART 3
RTE FORTRAN

INTRODUCTION

Regular FORTRAN and FORTRAN IV are segmented programs that execute in the background under control of RTE-II. The compilers consist of a main program and overlay segments, and reside in the protected area of the disc. At least 4K background disc-resident area is required to execute the regular FORTRAN compiler HP 28075; 5K for FORTRAN IV HP 24170; and 12K for FORTRAN IV HP 24177. Only one FORTRAN IV compiler can be used in the system at any one time.

RTE FORTRAN, a problem-oriented programming language translated by a compiler, is very similar to regular HP FORTRAN. Source programs, accepted from either an input device or disc LS tracks, are translated into relocatable object programs, and stored in the LG tracks of the disc and/or punched on paper tape. The object programs can be loaded by the RTE-II Relocating Loader and executed by an ON operator request. When a FORTRAN program has been completely debugged, the RTE-II Relocating Loader can make it a permanent part of the RTE-II System if desired.

FORTRAN REFERENCE

For a complete description of the regular HP FORTRAN Language, read the FORTRAN Programmer's Reference Manual (02116-9015). For a complete description of the HP FORTRAN IV Language, read the FORTRAN IV Programmer's Reference Manual (5951-1321).

COMPILER OPERATION

An RU, FTN operator request schedules the regular RTE FORTRAN compiler for execution. If FORTRAN IV is used, the operator request is RU, FTN4. All other parameters are the same. Before using RU, FTN, the operator must place the source program in the input device, or, if input is from a source file, specify the file location with an LS operator request. If planning to relocate and run, the operator allocates LG tracks with an LG operator request.

RU,FTN/FTN4

Purpose:

To schedule the FORTRAN compiler for operation.

Format:

RU, FTN, *input, list, punch, lines, 99*

or

RU, FTN4, *input, list, punch, lines, 99*

Where:

input = Logical unit number of input device. Use 2 for source file input from the disc (set to 5 if not given).

list = Logical unit number of list device (set to 6 if not given).

punch = Logical unit number of punch device (set to 4 if not given).

lines = Lines/page on listing (set to 56 if not given).

99 = The load-and-go parameter (LG command required first). If present, the object program is stored in the load-and-go tracks for later loading. Any punching requested still occurs. The 99 may occur anywhere in the parameter list, but terminates the list.

Example:

RU,FTN <is equivalent to RU,FTN,5,6,4,56>

MESSAGES TO OPERATOR

More than one source tape can be compiled into one FORTRAN program by leaving off the $END statement on all but the last source tape. When the end of each source tape is encountered (end-of-tape or EOT condition), RTE Driver DVR00 can interpret it in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how DVR00 subchannels were configured during generation. In any case, an EOT does not suspend the FORTRAN Compiler. Therefore, it is recommended that when compiling multiple tapes, DVR00 be configured to set the tape reader down on EOT (see the LU command). For more information refer to the DVR00 Manual (HP Part No. 29029-95001).

If an EOT causes the tape reader to be set down, the RTE-II system will output a message to the operator:

I/O ERR ET EQT #*eqt*

The operator must place the next source tape into the tape reader and set the tape reader up with the UP operator command.

UP,*eqt*

If an EOT does not cause the tape reader to be set down, the RTE-II system does not output any message and the compiler is not suspended.

At the end of the compilation (when the compiler detects the $END statement), the following message is printed.

$END,FTN

Two I/O error messages may be generated by the system when FTN attempts to write on the LG tracks (FTN is aborted).

IO06
IO09

IO06 means that the LG tracks were not defined by an LG operator request, and IO09 means that the LG tracks overflowed. The operator must define more LG tracks with LG and start compilation over again.

The compiler terminates abnormally if:

a. No source file is declared by LS, although logical unit 2 is given for input. Compiler error E-0019 (FTN2), or ERROR 05 (FTN4) is printed on the list device.

b. The symbol table overflows. Compiler error E-0014 (FTN2), or ERROR 03 (FTN4) is printed on the list device. $END, FTN does not appear after the error message using FTN2, but does appear when using FTN4.

FORTRAN FORMAT

The RTE FORTRAN Language is similar to the regular HP FORTRAN Language. The differences are described in the next few pages. RTE FORTRAN has additional capabilities, using EXEC calls. Read Section 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 = Punched binary tape (B not present does not affect binary output to load-and-go tracks).

L = List output.

A = Assembly listing.

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

T

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

n

n is a decimal digit (1 through 9) which specifies an error routine. The user must supply an error routine, ERR*n*. If this option does not appear, the standard library error routine, ERR0, 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, (type, pri, res, mult, hr, min, sec, msec)*

Where:

name is the name of the program (and its entry point).

type 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 = Real-Time Disc-Resident
3 = Background Disc-Resident
4 = Background Core-Resident
5 = Background Segment
6 = Illegal
7 = Library, utility
8 = If program is a main, it is deleted from the system

— or —

8 = If is a subroutine, then it is used to satisfy any external references during generation. However, it is not loaded in the relocatable library area of the disc.
9 = Foreground core-resident, uses background common
10 = Foreground disc-resident, uses background common
11 = Background disc-resident, uses foreground common
12 = Background core-resident, uses foreground common
13 = Background segment, uses foreground common
14 = Illegal

pri is the priority (1–32767, set to 99 if not given).

res is the resolution code.

mult is the execution multiple.

hr is hours.

min is minutes.

sec is seconds.

msec is tens of milliseconds.

COMMENTS

The parameters *type* through *msec* must appear in the order shown. And even though the parameters are optional, if any one parameter is given, those preceding it must appear also. For example:

PROGRAM*name*(,90)

is illegal and will be rejected by the system. The only method of legally defaulting the parameters is shown below:

PROGRAM *name*
PROGRAM *name*(3,90)

All parameters are set to 0 if not specified with the following two exceptions:

a. The priority parameter *pri* is set to 99, the lowest priority recognized by RTE FORTRAN.

b. The program type parameter *type* is set to 3 for a main program, or 7 for subroutines. Type 6 is illegal.

DATA STATEMENT

Purpose:

The DATA statement sets initial values for variables and array elements.

Format:

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

Where:

k is a list of variables and array elements separated by commas.

d is a list of constants (optionally signed) which can be immediately preceded by an integer constant (followed by an asterisk) identifying the number of times the constant is to be repeated.

/ is a separation, and is used to bind each constant list.

The elements of d_i are serially assigned to the elements of k_i, therefore, k_i and d_i must correspond one-to-one. If a list contains more than one entry, the entries must be separated by commas.

Elements of k_i may not be from COMMON.

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

Example:

```
DIMENSION A(3), I(2)

DATA A(1), A(2), A(3)/1.0,2.0,3.0/ ,
I(1), I(2)/ 2*1/
```

EXTERNAL STATEMENT

Purpose:

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

Format:

EXTERNAL $v_1, v_2, \ldots, v_5$

Where:

v_1 is the entry point of a function, subroutine, or library program, which exists externally.

Example:

```
FUNCTION RMX (X,Y,A,B)
RMX=X (A) * Y (B)
END
PROGRAM ABCDE
EXTERNAL XYZ, FL1
:
Z=Q-RMX (XYZ,FL1,3.56,4,75)
:
END
```

NOTE

If a library routine, such as SIN, is used as an EXTERNAL, the compiler changes the first letter of the entry point to "%." Special versions of the library routines exist with the first character changed to "%." See RTE Relocatable Library, Part 7 in this section.

PAUSE & STOP STATEMENTS

Purpose:

PAUSE provides a temporary program halt and the program to be suspended.

Format (as displayed):

name: PAUSE *oct numb*

Where:

name is the program name.

oct numb is the octal number given in the PAUSE. Note that the 'B' octal designator suffix is not required.

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

Purpose:

STOP causes the program to be terminated.

Format (as displayed):

name: STOP *oct numb*

Where:

name is the program name.

oct numb is the octal number given in STOP. Note that the 'B' octal designator suffix is not required.

ERRO LIBRARY ROUTINE

Purpose:

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

Format:

name: id type

Where:

name is the program name.

id is the routine identifier.

type is the error type.

COMMENTS

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.

PART 4
RTE ALGOL

INTRODUCTION

The RTE ALGOL compiler is a segmented program requiring 8K of background disc-resident area. The compiler accepts source programs written according to regular HP ALGOL with some additions and changes.

ALGOL REFERENCE

For a complete description of the HP ALGOL Language, including error messages, read the HP ALGOL Programmer's Reference Manual (HP Part No. 02116-9072).

COMPILER OPERATION

An RU,ALGOL operator request schedules the RTE ALGOL compiler for execution. Before using RU,ALGOL, the operator must place the source program in the input device, or, if input is from LS tracks, specify the file location with an LS operator request. If planning to relocate and run, the operator allocates LG tracks with an LG operator request.

RU,ALGOL

Purpose:

To schedule the ALGOL compiler for operation.

Format:

RU,ALGOL, *input, list, punch, lines, 99*

Where:

input = Logical unit number of input device. Use 2 for source file input from the disc. (Set to 5 if not given).

list = Logical unit number of list device (set to 6 if not given).

punch = Logical unit number of punch device (set to 4 if not given).

lines = Lines/page on listing (set to 56 if not given).

99 = The load-and-go parameter (LG command required first). If present, the object program is stored in the load-and-go tracks for later loading. Any punching requested still occurs. The 99 may occur anywhere in the parameter list, but terminates the list.

Example:

RU,ALGOL <is equivalent to RU,ALGOL,5,6,4,56>

MESSAGES TO OPERATOR

More than one source tape can be compiled into one ALGOL program by leaving off the END$ statement on all but the last source tape. When the end of each source tape

is encountered (end-of-tape or EOT condition), RTE Driver DVR00 can interpret it in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how DVR00 subchannels were configured during generation. In any case, an EOT does not suspend the ALGOL compiler. Therefore, it is recommended that when compiling multiple tapes, DVR00 be configured to set the tape reader down on EOT (see the LU command). For more information refer to the DVR00 Manual (HP Part No. 29029-95001).

If an EOT causes the tape reader to be set down, the RTE-II system will output a message to the operator:

I/O ERR ET EQT # *eqt*

The operator must place the next source tape into the tape reader and set the tape reader up with the UP operator command.

UP, *eqt*

If an EOT does not cause the tape reader to be set down, the RTE-II system does not output any message and the compiler is not suspended.

At the end of completion (when the compiler detects the END$ statement), the following message is printed.

$END ALGOL

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

NO SOURCE

is output to the system teleprinter and the compilation ceases.

Two I/O error messages may be generated by the system when ALGOL attempts to write on the load-and-go tracks (ALGOL is aborted).

IO06
IO09

IO06 means that the LG tracks were not defined by an LG operator request, and IO09 means that the LG tracks overflowed. The operator must define more LG tracks with LG and start compilation over again.

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

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

name: PAUSE *xxxx*

Where

name = the program name.

xxxx = number which has no significance.

Execution is then suspended. To restart the program, type

GO, *name*

See the GO operator command in Section II for a definition of the parameters.

ALGOL FORMAT

The first statement of an RTE ALGOL program is the HPAL control statement. The control statement does not use the symbol S (sense switch control). Also, after the NAM record-name, additional parameters may be specified.

ALGOL CONTROL STATEMENT

Purpose:

To define the output to be produced by the ALGOL compiler.

Format:

HPAL[,L,A,B,P],
"name"[,numb,type,pri,res,mult,hr,min,sec,msec]

Where:

L	=	Produce source program listing.
A	=	Produce object code listing.
B	=	Punch binary tape. B not present does not affect binary output to load-and-go tracks.
P	=	A procedure only is to be compiled.
name	=	Program name.
numb	=	A digit from 1 through 9 specifying the error-routine name. A library routine, ERR*numb* with *numb* = 1-9 must be supplied by the user. If this option is not specified, the error-routine name is ERR0. The error routine is called when an error occurs in the following routines: ALOG, SQRT, .RTOR, SIN, COS, .RTIO, EXP, .ITOI, TAN.
type	=	Program type.
pri	=	Priority.
res	=	Resolution code (0-4).
mult	=	Execution multiple (0-999).
hr	=	Hours (0-23).
min	=	Minutes (0-59).
sec	=	Seconds (0-59).
msec	=	Tens of milliseconds (0-99).

COMMENTS

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

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

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

The parameters *numb* through *msec* must appear in the order shown. And even though the parameters are optional, if any one paramenter is given, those preceding it must appear also. For example:

name , , , 90

is illegal and will be rejected by the system. The only method of legally defaulting the parameters is shown below:

name
or
name, 3, 3, 90

All parameters are set to 0 if not specified with the following two exceptions.

a. The priority parameter *pri* is set to 99, the lowest priority.
b. The program type parameter *type* is set to 3 if both *type* and P are not specified, or 7 if *type* is not specified and P is specified.

PART 5
RTE Assembler

INTRODUCTION

The RTE Assembler is a segmented program requiring 4K of background disc-resident area. The Assembler consists of a main program and segments, and resides in the protected system area of the disc.

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

The source tape passes through the input device only once, unless there is insufficient disc storage space. In this case, two passes are required. (See next page Messages To Operator.)

ASSEMBLER REFERENCE

For a complete description of the HP Assembler Language, read the Assembler Programmer's Reference Manual (HP Part No. 92060-90005).

ASSEMBLER OPERATION

An RU operator request schedules the RTE Assembler for execution. Before using RU,ASMB, the operator must place the source program in the input device, or if the input is from LS tracks, specify the file location with an LS operator request. If planning to relocate and run, the operator must allocate LG tracks with an LG operator request. The format for scheduling the Assembler is:

RU,ASMB

Purpose:

To schedule the Assembler for operation.

Format:

RU,ASMB, *input, list, punch, lines, 99*

Where:

input = Logical unit number of input device. Use 2 for source file input from the disc. (Set to 5 if not given.)

list = Logical unit number of list device (set to 6 if not given).

punch = Logical unit number of punch device (set to 4 if not given).

lines = Lines/page on listing (set to 56 if not given).

99 = Load-and-go parameter (LG command required first). If present, the object program is stored on the disc for loading, and any punching requested still occurs. The 99 may occur anywhere in the parameter list, but terminates the list.

Example:

RU,ASMB < is equivalent to RU,ASMB,5,6,4,56 >

MESSAGES TO OPERATOR

When a paper tape is being input through the tape reader, RTE Driver DVR00 can interpret and end-of-tape (EOT) in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how

DVR00 subchannels were configured during generation. In any case, an EOT does not suspend the Assembler. Therefore, it is recommended that when assembling multiple tapes, DVR00 be configured to set the tape reader down on EOT (see the LU command). For more information refer to the DVR00 Manual (HP Part No. 29029-95001).

If an EOT causes the tape reader to be set down, the RTE-II system will output a message to the operator:

I/O ERR ET EQT #*eqt*

The operator must up the tape reader with the UP operator command.

UP, *eqt*

If an EOT does not cause the tape reader to be set down, the RTE-II system does not output any message and the Assembler is not suspended.

At the end of assembly, the following message is printed:

$END ASMB

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

$END ASMB PASS

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

GO,ASMB

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

$END ASMB CS

The current assembly aborts.

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

$END ASMB XEND

The current assembly aborts.

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

$END ASMB NPRG

The current assembly aborts.

RTE-II generates two messages when ASMB attempts to write on the LG tracks (ASMB is aborted).

IO06
IO09

IO06 means that the load-and-go tracks were not defined by an LG operator request, and IO09 means that the LG tracks have overflowed. The operator must define more LG tracks with LG and start compilation over again.

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

#*tape numb*

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

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

PG *page numb*

page numb is the page number (in the listing) of the previous error diagnostic.

PG 000 is associated with the first error in the program.

These messages occur on a separate line, above each error diagnostic in the listing.

ASSEMBLER CONTROL STATEMENT

The control statement has the same form as that of regular Assembler Language; and although only relocatable code can be run under RTE, the RTE Assembler accepts and assembles absolute code. Absolute code is never stored in the LG tracks. To get absolute code, the control statement must include an "A." The "R", however, is not required for relocatable code. An "X" causes the assembler to generate non-extended arithmetic unit (non-EAU) code. B is required to punch a binary tape. B not present does not affect binary output to LG tracks.

The memory protect feature, which protects the resident executive from alteration (except in the case of privileged library routines), interrupts the execution of a user program under these conditions:

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

When an interrupt occurs, memory protect gives control to the system which checks to see if the interrupt was from a legal system call. If not, the user program is either suspended or aborted (depending on bit 15).

NAM STATEMENT

Purpose:

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

Format:

Nam *name, type, pri, res, mult, hr, min, sec, msec, id*

Where:

name is the name of the program.

type is the program type (set to 0 if not given):

0 = System program
1 = Real-time core-resident
2 = Real-time disc-resident
3 = Background disc-resident
4 = Background core-resident
5 = Background segment
6 = Library (re-entrant or privileged)
7 = Library, utility
8 = If program is a main, it is deleted from the system

– or –

8 = If program is a subroutine, then it is used to satisfy any external references during generation. However, it is not loaded in the relocatable library area of the disc.
9 = Foreground core-resident, uses background common
10 = Foreground disc-resident, uses background common
11 = Background disc-resident, uses foreground common
12 = Background core-resident, uses foreground common
13 = Background segment, uses foreground common
14 = Library, core resident

pri is the priority (1 to 32767, set to 99 if not given).

res is the resolution code
mult is the execution multiple.
hr is hours.
min is minutes.
sec is seconds.
msec is tens of milliseconds.

(Time values, set to 0 if not given. See Section II, IT, for meaning)

id comments field-separated from parameters by a space.

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

COMMENTS

The parameters of the NAM statement, beginning with *type* and ending with *msec*, are separated by commas. A blank space within the parameter field will terminate that field and cause the Assembler to recognize the next entry as the comment field (*id*). The first parameter must be separated from the program *name* by a comma. The parameters are optional, but to specify any particular parameter, those preceding it must also be specified.

The comment field (*id*) can be a maximum of 73 characters due to the restriction of the source statement size. The source statement will be truncated after column 80.

The comment field in the NAM statement will be included as ASCII in the relocatable binary object code. This means that when the program is relocated with the RTE loader, the comments field will be printed out as part of the NAM statement.

CREATING TYPE 4 ENTRY RECORDS

The user can create type 4 entry records using the RTE Assembler RPL instruction. When an entry point is RP'd with a code replacement value (e.g., .FAD to 105000) the RTE Loader intercepts the entire JSB instruction and substitutes the RP'd value in its place. This means the user can eliminate software subroutines by replacing their entry points with microcode instructions. It also allows the user to expand the system by adding programs referencing entry points to subroutines that were not loaded during generation.

Entry points are replaced using the RTE Assembler in the following manner:

a. Create a source file (e.g. punch a paper tape or create a disc file) using the following format example:

label	*operation*	*operand*
.FAD	RPL	105000B

b. Assemble the source file to obtain a relocatable and load the relocatable file into the LG track area.

c. Load your program with the RTE Loader.

PART 6
RTE-II Loader

INTRODUCTION

The RTE-II On-Line Relocating Loader provides a means for linking relocatable files produced by compilers or assemblers together with one or more library files. The resulting program can optionally be loaded into the system and run; or the program can be loaded into the background with the DEBUG library routine linked to it.

The RTE-II Loader has the following features:

- Can operate under control of the File Manager operating under Batch mode.
- Is swappable and can be operated in either background or foreground disc-resident areas.
- Allows programs to reference common areas away from program location.
- Loads referenced library routines from LG tracks.
- Can force the relocation of subroutines which have not been referenced by a previously relocated module. For example, you can force the relocation of your own version of a subroutine that already exists in the system library (see *library* parameter, GO,LOADR command).
- Allows a program to be permanently added to the system (i.e., only the leader can be used to purge a permanent program; the OF, *name*, 8 command will not remove a permanent program from the system).
- Allows programs to reference absolute and code replacement type ENT records.
- Uses system area tracks that have been vacated by deleted programs.
- Allows a program to be temporarily loaded into the foreground area.

  Uses the "short" ID segment (when available) when loading a background program segment. Also, does not restrict the arrangement of subroutines following segments. For example, if a subroutine is shared by two segments, only one copy of it is necessary on the load-and-go tracks.

LG TRACK AREA

The RTE system provides facilities for the assembly or compilation, relocation, and scheduling for execution of a user program without intervening paper tapes. To accomplish this, the assembler or compiler accepts source statements from which it generates relocatable object code. The relocatable code may be stored on disc in predefined LG tracks (see LG Operator Request, Section II). Then, separate operator requests initiate relocation (e.g., RU,LOADR) and schedule execution (e.g. ON, or RU, *program name*).

Two rules should be remembered when using the LG track area:

a. Do not reset the LG track area using the LG command if the LG track area was just used for a forced relocation and additional library routines remain to be relocated. To do so would result in the loss of the additional routines to the loader.

b. When the initial input is from the LG track area, the *input option* parameter of the GO,LOADR command is set to 99 to indicate that the LG tracks have been reset with new data moved into them. Setting the *input option* to 2 implies that the LG tracks have not been reset and additional data has been added to them.

When using the loader, the programmer can structure the LG track area with a single main program and subroutines, or with a main program and segments. For relocating a main program with segments from the LG track area, some constraints are imposed by LOADR. These constraints are:

a. Once the LG track area has been scanned and relocated, there should be no undefined external

references in the main program. If there are, they will be listed after the last segment is relocated but they cannot be satisfied.

b. A segment cannot satisfy any external references made by another segment. However, the main program can satisfy segment external references, and segments can satisfy main program external references.

When the loader terminates either with the message /LOADER: SEND, or /LOADR ABORTED, the LG track area is cleared. If the loader terminates in some other manner, the LG track area is not cleared.

BACKGROUND LOADING

During loading, the programs are relocated to the start of the background disc-resident area and linked to external references such as EXEC, the resident library, or the relocatable library. Any segments overlay the core area following the main program and its subroutines.

Regardless of the program type recorded in the NAM record, the third parameter of the RU,LOADR request indicates whether the program is to be set up for foreground or background operation. Default results in the latter and the program is not permanently set up in the system (i.e., the program is lost on boot-up). The DEBUG library subroutine can be linked to the program if desired.

The loader stores the absolute version of the program, its subroutines, and linkages on a disc track or a group of contiguous tracks; it then assigns the disc tracks to the System (i.e., not available as scratch or data tracks by programs) and updates, in core only, the ID segment assigned to the program. The program and its subroutines may be as large as the background disc-resident area. Common area can be allocated in one of several areas according to the needs of the programmer. Refer to the optional parameters in the RU,LOADR command.

ON–LINE MODIFICATION

Using the loader, the operator can permanently modify the set of disc-resident user programs in a configured RTE-II System. The loader adds new disc-resident real-time or background programs, and replaces disc-resident programs with updated versions that have the same name. When a program is being replaced it must be dormant, not in the time list, and have a zero point of suspension. The OF operator request deletes those disc-resident programs loaded temporarily into the system by the loader. The OF operator request will not delete program segments that were permanently added on-line or stored during generation.

When the system is generated, RTGEN, the system generator, stores the programs on the disc in an absolute, packed format. Each main program is identified and located by a 28-word identification segment (ID segment) of which there is one copy in core and one on the disc. For disc-resident programs, the program's disc location as well as its core memory bounds, is kept in the ID segment. When a main program and its segments are loaded, the segments are identified and located by a 9-word "short" ID segment. Refer to Appendix A for the ID segment formats.

RTGEN can create a number of blank 28-word and 9-word ID segments so that the loader can add new programs and segments to the permanent system later. The addition or replacement of a program involves the conversion of relocatable programs into an absolute unit, finding space on the disc to store it, and recording information in the ID segment. The loader always attempts to use the "short" ID segment for identifying a program segment. However, if a "short" ID segment is not available, a regular 28-word one is used.

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

Core-resident programs cannot be replaced because the length of the program and linkage area is not kept in the ID segment for core-resident programs, nor can they be added because this would require changing the disc-resident program area origins.

If a user supplied routine is to be referenced by a program, and a core resident subroutine of the same name already exists in the system, then the user supplied routine must be loaded before any reference is made to it. Conversely, if a system relocatable library routine has to be replaced, the user supplied routine can be force loaded even after it has been referenced.

LIMITATIONS

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

a. A common length exceeds the original common block. The length of local common is governed by the first relocatable module encountered by the loader.

b. The base page linkages exceed the corresponding linkage area (linkage area for background/foreground disc resident programs) as established during system generation.
c. The length of the absolute program unit exceeds the area available.
d. Disc space is not abvailable to store the program.
e. A blank ID segment is not available for adding a program. (A program previously added could be deleted to provide a blank ID segment.)

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

SEGMENTED BACKGROUND PROGRAMS

Segmented programs can be added and replaced in any order as long as the main program is always entered first. Permanent addition of main segment programs need not necessarily result in the main and segments being stored on contiguous tracks.

When replacing segmented programs that were incorporated into the sytem at generation time, the operator must replace every segment with a new segment having the same name, or remove the segment permanently from the system. Additional extra segments, however, may be added in a replacement and any segments left over (from the old program) as a result of replacement may be deleted using the loader.

NEW PROGRAM ADDITION

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

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

PROGRAM REPLACEMENT

In a replacement, if the new program can fit in the disc area of the old program (both programs must have the same name), the new program uses the ID segment of the old. The new program is generated onto temporary tracks, and then, if it can fit in the old area, or within another area within the system gained as a result of deleting a program incorporated during generation, it is transferred. If not, the temporary tracks attain system track status and a blank ID segment is assigned to it. The old ID segment is blanked but retains its disc space for later use by another program.

PROGRAM DELETION

A temporary program is deleted from the system with the OF,*name*,8 command. A permanent program (which is defined as a program loaded during generation, or on-line with the loader as a permanent or "edit" load), is deleted with the loader. When using the loader to delete a permanent program, the first two paramenters are defaulted, and the *opcode* parameter is set to 4.

This blanks the program's ID segment making it available for loading another program. The tracks (unless they are system tracks) containing the program are released. If the program had been saved on FMP tracks through the File Manager, those tracks are not released to the system but remain as FMP tracks.

COMMON ALLOCATIONS

There are three options the user can specify when allocating common area for a program.

SYSTEM COMMON – This implies a background program having its common in the background system common area, or a foreground program having its common in the foreground common area.

LOCAL COMMON – Common area for a background program is established directly above the background resident area, or for a foreground program directly above the real-time resident area. The common area will be swapped during program execution.

REVERSE COMMON – This implies a background program having its common in the foreground system common area. Note that improper programming can cause memory protect violations. Conversely, a foreground program can use and reference the background system common area.

LOADER OPERATION

The operator schedules the loader for execution with the RU operator command.

RU,LOADR

Purpose:

To relocate and load programs so they can be scheduled by an ON or RU operator request.

Format:

RU,LOADR , *input, list, opcode, fmt, listing*

Where:

input = Logical unit number of input device. If set to 99, the load-and-go tracks are used, but 99 does not terminate the parameter list (default = 5).

list = Logical unit number of list device (default = 6).

opcode = Operation Code:

Most Significant Digit			Least Significant Digit	Loader Operation
System	Local	Reverse		
Common Area				
1	*2	3	0	BG temp.
1	*2	3	1	BG w/DEBUG
*1	2	3	2	On-line Edit
—	—	—	3	List Programs
—	—	—	4	Purge Program
1	*2	3	5	FG temp.
*1	2	3	6	FG replace
*1	2	3	7	FG add new
*1	2	3	8	BG replace
*1	2	3	9	BG add new

NOTE

The asterisks indicate upper digit default values.

fmt = Format of program.

0 = Single main program and subroutines.
1 = Main and segments.

listing = Listing parameter (only when *input* = 99).

0 = List program name, bounds, and entry points.
1 = List only entry points.
2 = List only program name and bounds.
3 = Omit program name, bounds, and list of entry points from listing.

COMMENTS

When the program is stored on the disc, any BSS location will contain NOP instructions (i.e., zero). "WAITING FOR DISC SPACE" is printed when a track allocation cannot be made. The loader repeats the disc request and is suspended until space becomes available.

If a single program is loaded and it is not necessary to re-schedule the loader with the GO command, the loader terminates with the following message:

/LOADR: *name* READY
/LOADER: $END

Where

name is the name of the main user program. The loader terminates and the program is ready to run.

opcode PARAMETER

The most significant digit of the *opcode* parameter indicates the type of "common" allocation desired and the least significant digit what operation to perform. If the upper digit is not supplied, the type of "common" allocation defaulted to depends upon the type of operation code. The asterisks indicate these defaults.

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

Setting *opcode* = 2 initiates the loader for an on-line edit which requires additional information through the GO. LOADR command. Refer to the ON-LINE LOADER OPERATION heading for additional information.

Setting *opcode* = 3 causes a listing of all the programs and blank ID segments to be printed. For each ID segment in the system the following format is used.

name, type, priority

name is the program name.

type is the program type:
1 – Real-time resident.
2 – Real-time disc-resident.
3 – Background disc-resident.
4 – Background resident.
5 – Background segment.

priority is the program priority, from 1 to 32767.

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

<LONG BLANK ID>

– or –

<SHORT BLANK ID>

The loader terminates after the list is complete.

Setting *opcode* = 4 initiates the loader to purge a permanent program. Using this option, the *input* and *list* parameters automatically default to 1. The loader's response to this option is:

/LOADR: PNAME?

Enter the program's name on the keyboard input device and it will be permanently removed from the system. To abort the command (and loader) enter /A.

Setting *opcode* = 5-9 allows the loader to run in Batch mode without operator interaction.

fmt PARAMETER

If *input* = 99 (specifying load-and-go tracks), and *fmt* = 1 (main and segments), an automatic segmented program load is done and the loader immediately scans the library for entry points. The LG tracks must contain the main and its subroutines followed by a segment and its subroutine. Where the same subroutine is required by both the main and segment, but is not in the library, it need only appear in the main. Subroutines required by more than one segment, but not the main, can appear with each segment in the LG tracks for greater speed in loading. Or, if the user desires, only one copy of the subroutine can be placed in the LG tracks and the tracks scanned as a library (loading time is somewhat greater). Note that if the loader suspends as a result of undefined externals in any segment, the user may move additional subroutines to the end of the LG tracks and scan the area as a library, or have the loader scan the input device for required subroutines. If there are undefined externals in the main, the loader prints the message MAIN followed by UNDEFINED EXTS and then suspends. It is not possible to satisfy these undefineds using the loader. The only command acceptable is GO,4 (continue loading without fulfilling externals).

If *fmt* = 0, a single main program and subroutines will be merged into an absolute program unit. To load another main program, the loader must be scheduled again.

LOADING THE BINARY CODE

For a main/segment load, the main program must be entered first to establish the segment area boundaries. The library must be scanned (GO,LOADR,1) after each main program and segment (except the last segement).

The loader scans the relocatable programs and subroutines as it reads them in, keeping track of any external references. If input is initially from the disc as specified by 99 in the RU statement, the loader immediately scans the library for entry points. If input is from paper tape and DVR00 has been configured to set the tape reader down on EOT, the loader suspends with the message:

I/O ERROR ET EQT *#eqt*
/LOADER: LOAD

EQT *#eqt* is unavailable until the operator reinitializes it.

UP, *eqt*

If an EOT does not cause the tape reader to be set down, the RTE System does not output any message but does suspend the loader.

LOADER RESCHEDULING

The operator reschedules the loader with the GO operator command.

GO,LOADR (Background)

Purpose:

To reschedule the loader to continue program loading.

Format:

GO,LOADR, *input option, entry pts, library*

Where:

input option = 0, 2, or 99 indicates a program load if *library* is 0 or not entered.
= 0 – Load from the binary input unit (LU5).
= 1 – Scan disc resident relocatable library and load referenced library routines.
= 2 – Load from LG tracks. LG tracks have been loaded from previously.
= 3 – Load from the relocatable library for the last segment in a main/segment load.
= 4 – Continue without loading any remaining referenced library routines.
= 98 – List undefined externals.
= 99 – Load from LG tracks for the first time.
= *lu* – Where *lu* is a logical unit number and not one of the above numbers.

entry pts = 0 – List entry points.
= 1 – Omit list of entry points

library = 0 – Load all data (force load).
= 1 – Satisfy undefined externals only (library scan).

COMMENTS

If the LG tracks are read with a GO,LOADR,99 (or 2) command, the LG track area is not cleared. However, the RU,LOADR,99 command will still clear the load-and-go tracks upon a successful load.

input option **PARAMETER**

- Once you have force loaded from the load-and-go area, subsequent use of the load-and-go area requires that *input option* = 2.
- After the loader encounters the last segment during a main/segment load from the LG track area, no more segments can be read in (even from the binary input device).
- If undefined externals remain in the main of a segmented program (which the loader discovers after loading the last segment) and the message "MAIN–UNDEFINED EXTS" is printed, the undefined externals cannot be satisfied. The only legal response at that time is GO, LOADR, 98 or 4.
- It is permissible to use the LG track area more than once with GO, LOADR, 99 without resetting it with the LG command as long as the last load was not a force load.

library **PARAMETER**

The *library* parameter does not apply when *input option* = 1, 3, 4 or 98. If *library* = 1 then library input from the specified source is assumed. If *library* = 0 then the input is force loaded. If the library is read through an input device (not from the LG track area) and the message LOAD LIB is printed, the library must be scanned again to satisfy an undefined external. When the message LOAD is printed, the library scan is finished.

MATCHING EXTERNALS

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

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

UNDEFINED EXTS

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

To load additional programs from the input unit, the operator types:

GO,LOADR

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

GO,LOADR,4

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

**name address*

Where

name is the entry point name, and

address is its absolute location in octal.

END OF LOADING

At the end of a normal load, or after loading the last segment of a main/segment load, the loader prints the following message and terminates itself.

/LOADR: *name* READY
/LOADR: $END

Where

name is the name of the main user program. The loader terminates and the program is ready to run.

After loading a main or segment of a main-segment load (end-of-tape mark) the loader prints the following message and waits for the GO,LOADR entry for the next segment.

/LOADR: LOAD

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

GO,LOADR,3

The loader proceeds to the end of loading, as described above.

The operator can schedule the program for execution by an ON or RU operator request (see Section II). The disc tracks containing the program are assigned to the system and are software-protected. The program, if a temporary load, can be eliminated from the system with the OF operator command, or if a permanent load can be eliminated with the RU,LOADR,,,4 command.

LOADER OPERATION (ON-LINE EDIT)

The operator schedules the loader for on-line edit operations by setting *opcode* = 2 in the RU,LOADR command. Refer to the RU,LOADR command for more details.

RU,LOADR,*input,list,2,fmt,listing*

The loader requires additional information to carry out the modifications so it prints the following message and suspends.

/LOADR: "GO" WITH EDIT PARAMETERS

The operator must disable the hardware disc protect switch and reschedule the loader.

LOADER RESCHEDULING (ON-LINE EDIT)

The operator reschedules the loader with the GO operator command. This GO,LOADR command for on-line edit operations contains an additional command.

GO, LOADR (On-Line Edit)

Purpose:

To reschedule the loader to input additional parameters required for the edit operation.

Format:

GO,LOADR, *operation, prog type* [,*priority*]

Where:

operation = 1 for an addition operation.

= 2 for a replacement operation (program must be dormant).

prog type = 2 for a real-time disc-resident program, or

= 3 for a background disc-resident program.

priority = priority (optional) form 0 to 32767 (a 0 means use the priority value in the NAM record of the program or, if that priority is 0, use 32767).

COMMENTS

Any errors cause the error message "L10" to be printed. The disc hardware protect must be disabled before the program is loaded, then re-enabled after loading.

When the GO request is entered, the loader proceeds to load the program, keeping track of any external references. If input is from the disc as specified by 99 in the ON statement, the loader immediately scans the library for entry points. If input is from paper tape, the loader suspends with the message.

I/O ERR ET EQT *#eqt*
/LOADER:LOAD

EQT *#eqt* is unavailable (see DN, Section II) until the operator declares it up.

UP,*eqt*

The operator reschedules the loader with the GO request exactly as described previously under GO,LOADR (Background).

RTE DEBUG LIBRARY SUBROUTINE

DEBUG, a utility subroutine of the RTE Relocatable Library is appended to the user's program by the loader when the *opcode* parameter in the RU,LOADR command is set to 1, and allows programs to be debugged (i.e., checked for logical errors on-line) during execution. Programs that expect starting parameters or that call RMPAR (see Section III, Program Suspend Exec Call) cannot use DEBUG because DEBUG uses the parameters.

After the user's program is loaded with DEBUG appended to it, the user turns his program on with the *input* parameter set to 1 (keyboard input).

RU,*name*,1

Where

name is the name of the user's program.

The primary entry point of the user's program (the location where execution begins) is set to DEBUG so that when the program is turned on with an RU operator request, DEBUG takes control and prints a message:

BEGIN 'DEBUG' OPERATION

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

ENTRY ERROR

For further details on the uses of DEBUG, refer to the Pocket Guide to Hewlett-Packard Computers (HP Part No. 5951-4423) BCS Section, where the non–RTE DEBUG routine is described.

The following commands describe DEBUG operations.

B,A	Instruction breakpoint at address A. (NOTE: if A = JSB EXEC, a memory protect violation occurs.)
D,A,N1[,N2]	ASCII dump of core address N1 or from N1 to N2.
D,B,N1[,N2]	Binary dump of core address N1 or from N1 to N2.
M,A	Sets absolute base of relocatable program unit.

R,A	Execute user program starting at A. Execute starting at next location in user program (used after a breakpoint or to initiate the program at the transfer point in the user program).
S,A1,D1	Set D1 in location A1.
S,A1,D1 . . . D*n*	Set D1 to D*n* in successive memory locations beginning at location A1.
W,A,D1	Set A-Register to D1.
W,B,D2	Set B-Register to D2.
W,E,D3	Set E-Register (0 = off, non-zero = on).
W,O,D4	Set Overflow (0 = off, non-zero = on).
X,A	Clear breakpoint at address A.
A	Abort DEBUG operation.

RELOCATING LOADER ERROR MESSAGES

Messages are printed in this format:

/LOADR: *message*

"L" ERROR MESSAGES

L01 – Checksum error

L02 – Illegal record

These errors are recoverable (except in Batch mode). The offending record can be reread by repositioning the tape and typing.

GO,LOADR

L03 – Memory overflow.

L04 – Base page linkage area overflow.

L05 – Symbol table area overflow.

L06 – Common block error.

a. Exceeding allocation in a replacement or addition.

b. In a normal background load, first program did not declare largest common block.

L07 – Duplicate entry points.

L08 – No transfer address (main program) in the program unit. Another program may be entered with a GO operator request. (This also occurs when load-and-go is specified, but no program exists in the load-and-go area.)

L09 – Record out of sequence.

L10 – Operator request parameter error. GO requests may be retyped; RU requests may not.

L11 – Operator attempted to replace or purge a core-resident program.

L12 – LG area used without presetting (*input option* = 2 IN 'GO'). *input option* was not input as 99 previously.

L13 – LG area has been illegally reset – overwritten. Program addition on LG area not allowed if it has already been specified for program input, or LG area was once used for force loading with *input option* = 99 and it is again being used with *input option* = 99.

L14 – ASMB produced illegal relocatable. A DBL record refers to an external which has not been defined (the original can not be found in the symbol table).

L15 Forward reference to a type 3 of type 4 ENT or to an EXT with offset which has not yet been defined.

ADDITIONAL MESSAGES

NO BLANK ID SEGMENTS

This message is printed when available (i.e., blank) ID segment is not found. The loader calls for program suspension. The operator may then delete a program from the system (OF operator request) or may terminate the loader.

DUPLICATE PROG NAME – *name*

This message is printed when a program name is already defined in the system for a normal load or a program addition. The loader changes the name of the current program

by replacing the first two characters with periods (e.g., JIMB1 becomes ..MB1). The second duplicate program name aborts the loader.

WAITING FOR DISC SPACE

This message is printed when a track allocation cannot be made. The loader repeats the disc request and is suspended until space becomes available.

UNDEFINED EXTS

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

LOAD

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

SET PRGM INACTIVE

This message is printed if the loader attempts to replace a program that is not dormant, is in the time list, or has a non-zero point of suspension. The program to be replaced must be set inactive using the OF operator request.

LOAD LIB

This message is printed when an end-of-tape condition is detected from the input device being used for library input and the loader needs the library to be scanned again.

PART 7
RTE Relocatable Library

INTRODUCTION

There are three libraries or collections of relocatable subroutines that can be used by RTE-II, the RTE/DOS Relocatable Library, EQU version (F2E. *n*), and the RTE/DOS FORTRAN IV Library (F4D. *n*), and the RTE/DOS Floating Point Library (F2F. *n*). Refer to Figure 4-1. 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 (HP Part No. 02116-91780).

In addition to the above libraries there is a system library (92001-16005) which contains routines unique to the RTE-II system. These routines are documented in this section.

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-II through entry point $LIBR to save the data. At the end of execution, the re-entrant routine calls RTE-II 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 the system 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

```
        NAM   xxxxx,6
        EXT   $LIBR, $LIBX

ENTRY   NOP           Entry point of routine.
        JSB   $LIBR   Call RTE-II to save temporary
                      data.
        DEF   TDB     Address of temporary data.
        :
        ,             Program instructions
        ,
EXIT    JSB   $LIBX   Call RTE-II to restore data.
        DEF   TDB
        DEC   m       m is for routines with two return
                      points in the calling program; 0
                      specifies the error-print return
                      and 1 the normal return. For rou-
                      tines with only one return point,
                      m = 0.
TDB     NOP           System control word.
        DEC   n+3     Total length of current block.
        NOP           Return address to calling program.
```

T1
.
.
.
Tn

} Temporary data (*n* words).

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.

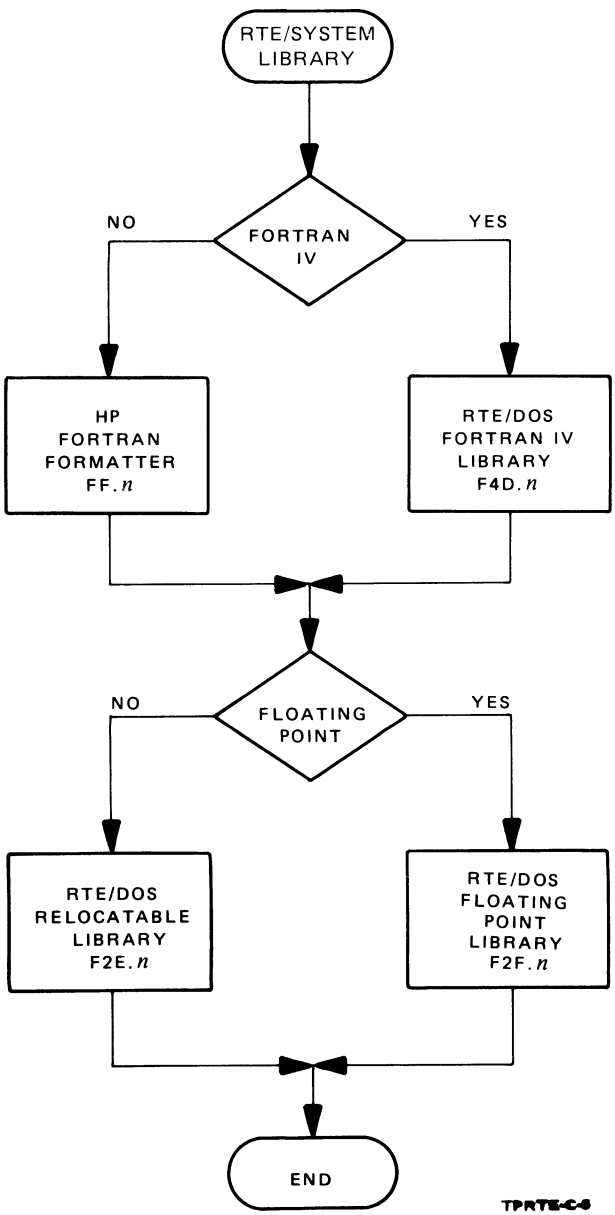


Figure 4-1. RTE Library Configuration Diagram

FORMAT OF PRIVILEGED ROUTINE

```
      NAM xxxx,6
      EXT $LIBR,$LIBX

ENTRY NOP          Entry points to the routine.
      JSB $LIBR    Call RTE-II to disable the
                   interrupt system and memory
      NOP          protect fence.
       .
       .
       .
EXIT  JSB $LIBX    Call RTE-II to return to calling
                   program and enable interrupts
                   and memory protect fence.
      DEF ENTRY    Location of return address.
```

Re-entrant and privileged routines may be placed in the resident library during generation by either of the following methods.

- If the routine is declared as an external (called) by a resident (type 1 or 4) program, or is called by another resident library routine, the routine will automatically be placed in the resident library by the generator.

- The routine can be changed during the parameter input phase of generation to a type 14 routine (it also could have been assembled as a type 14).

Note that type 6 routines not put in the core resident library are changed to utility routines (i.e., type 7).

UTILITY SUBROUTINE STRUCTURE

Utility subroutines are subroutines which cannot be shared by several programs because of integral design or I/O operations. A copy of the utility routine is appended to every program that calls for it. The library subroutine FRMTR (FF.n) or FNTIO (F4D.n) which carries out FORTRAN I/O operations, and the PAUSE subroutine are examples of utility routines.

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

RE-ENTRANT I/O

A re-entrant subroutine may do I/O using the standard EXEC requests. If the buffer is in the temporary data block (TDB) of either itself or another re-entrant routine that called it, the calling program is swappable. If the buffer is in the user area the program is not swappable. (i.e., if the buffer is not in the TDB or user common area the program is not swappable).

A subroutine called REIO is furnished to allow the user to do re-entrant I/O. REIO is a utility type library routine that has within its structure a re-entrant routine. Therefore, the routine may not be put in the resident library, it must be appended to each program that calls it.

The calling sequence for REIO is:

```
CALL REIO(ICODE,ICNWD,IBUFR,IBUFL)
```

where the parameters are as described in the READ/WRITE EXEC call in Section III of this manual. REIO will always do the requested I/O; however, it will do re-entrant I/O only if the buffer is less than 130 words (to save system memory), and the buffer address is at least three words above the current fence address. If the sign bit is set on ICODE the same error options available with the EXEC Call are effected (i.e., error return followed by normal return). REIO returns the same values in the A- and B-Registers as the standard EXEC Call.

OTHER SUBROUTINES

Several other subroutines are provided for programming convenience. The basic calling sequences are provided here for reference.

BINRY

FORTRAN programs can call BINRY, the disc read/write library subroutine, to transfer information to or from the disc. The call must specify a buffer array, the array length in words, the disc logical unit number, track number, sector number, and offset in words within the sector. (If the offset equals 0, the transfer begins on the sector boundary; if the offset equals *n*, then the transfer skips *n* words into the sector before starting.) BINRY has two entry points: BREAD for read operations and BWRIT for write operations.

For example,

```
CALL BWRIT (ARRAY,N,IDISC,ITRK,ISECT,IOFST)
CALL BREAD (ARRAY,N,IDISC,ITRK,ISECT,IOFST)
```

Where:

ARRAY	=	Address of the first element
N	=	Number of words
IDISC	=	Disc logical unit number
ITRK	=	Starting track number
ISECT	=	Starting sector number
IOFST	=	Number of words offset within a sector

There are three basic ways that data can be written on the disc in relation to sector boundaries. Care must be used in planning the WRITE statement in two of the cases to avoid losing existing data.

One form of writing data on the disc is offset=*n* (i.e., transfer begins within a sector), and less than the sector is written, or the data transfer ends on a sector boundary. The entire first sector is initially read into an internal buffer, the data is modified according the BWRIT statement, and then the entire sector is rewritten on the disc with no data loss. No special precautions are required in this instance.

A second form of writing data on the disc is offset=0 (i.e., transfer begins on a sector boundary), and less than the sector is written. The remaining data in the sector will be lost if the following precaution is not taken. The entire existing sector on the disc can first be read into a user's buffer, modified to reflect the desired changes, and then rewritten on the disc as a full sector.

A third form of writing data on the disc is offset=0 or *n*, and a sector boundary is crossed in the data transfer. The remaining data in the final sector will be lost if the following precaution is not taken. The entire final sector (of the data transfer) on the disc should be read into a user's buffer, modified to reflect the desired changes, and then rewritten on the disc as a full sector.

PARSE SUBROUTINE

The following subroutine is used to parse an ASCII string. Note that the calling subroutine must be privileged when using the following Assembly Language calling sequence:

```
LDA BUFAD  buffer address
LDB CCOUN  character count
EXT $PARS
JSB $PARS
DEF RBUF
 -return-
```

Where RBUF is 33 words long. The result of the parse of the ASCII string at BUFAD is stored in RBUF using 4 words per parameter and are set as follows:

Word	Entry	
1	FLAG WORD	0 = NULL 1 = NUMERIC 2 = ASCII
2	VALUE(1)	0 If NULL; Value if Numeric; first 2 characters if ASCII.

3	VALUE(2)	0 If NULL or numeric else the 3rd and 4th characters.
4	VALUE(3)	0 If NULL or numeric else the 5th and 6th characters.

ASCII parameters are separated from numeric parameters by examination of each character. One or more non-digit characters (except a trailing "B" or leading "–") makes a parameter ASCII.

The 33rd word of RBUF will be set to the number of parameters in the string.

The Parse routine ignores all blanks and uses commas to delimit parameters. ASCII parameters are padded to 6 characters with blanks or, if more than 6 characters, the left most 6 are kept. Numbers may be negative (leading "–") and/or octal (trailing "B").

A FORTRAN callable interface to $PARS is provided with the calling sequence as follows:

```
CALL PARSE (IBUFA,ICONN,IRBUF)
```

Where the parameters are as described for the Assembly Language call above.

BINARY TO ASCII CONVERSION SUBROUTINES

The following subroutine is used to convert an integer binary number to ASCII. Note that the calling program must be privileged when using. The Assembly Language calling sequence:

```
LDA numb
CLE or CCE (see text)
EXT $CVT3
JSB $CVT3
 -return-
E=1,
A=address of result
B=value at invocation
```

$CVT3 converts the binary number in the A-Register to ASCII, suppressing leading zeros, in either OCTAL (E = 0) or decimal (E = 1). On return, the A-Register contains the address of a three word array containing the resultant ASCII string.

$CVT1 has the same calling sequence as $CVT3 except that on return, the A-Register contains the least two characters of the converted number.

A FORTRAN callable interface to $CVT3 is provided with the calling sequence as follows:

```
(decimal) CALL CNUMD (binary numb,addr)
(octal)   CALL CNUMO (binary numb,addr)
```

where *binary numb* is the binary number to be converted and *addr* is the address where a three word array (6 ASCII characters) begins. Leading zeros are suppressed.

The following subroutine converts a variable to ASCII base 10 and returns the least two digits in "I". The FORTRAN calling sequence is:

```
I=KCVT(J)
```

MESSAGE PROCESSOR INTERFACE

The message processor processes all system commands, see Section II. A FORTRAN call to the system message processor is provided with the calling sequence as follows:

```
I = MESSS (IBUFA,ICOUN,LU)
```

Where IBUFA contains the ASCII command, ICOUN is an integer containing the character count, and LU is optional.

The value on return will be zero if there is no response or the negative of the character count, if there is a message. The message, if any, will be in IBUFA.

If the request is RU or ON (starting in 1st column) and the first parameter is zero or absent, then the first parameter will be replaced by LU. LU is optional. If it is not supplied, no action takes place.

INTERRUPTING LU QUERY

A calling sequence is provided to find the logical unit number of an interrupting device from the address of word four of its equipment table entry. The address of word four is placed in the B-Register by the driver and used in the following sequence:

```
LDB EQT4 (done by DVR00 and DVR65)
```

Not necessary if address of EQT4 has already been placed into 'B' by driver, or by another program/subroutine.

```
EXT EQLU
JSB EQLU
DEF *+2 or *+1
DEF LUSDI
```

EQLU will return with:

A-Register = 0 if an LU referring to the EQT was not found.

= LU if the LU was found.

B-Register = ASCII "00" or the LU number in ASCII e.g., "16."

LUSDI = (optional parameter) value is returned to this parameter, as well as in the A-Register.

Other variations of the call are (passed from DVR00 or DVR65):

```
EXT EQLU
JSB EQLU
DEF *+1
STA LU
STB ASCLU
-or-
LU=EQLU (LU)
```

PARAMETER RETURN SUBROUTINES

There are two routines used to pass parameters to the program that scheduled the caller with wait. The scheduling program may recover these parameters with RMPAR.

The first routine is called PRTN, passes five parameters, and clears the wait flag. This means that the caller should terminate immediately after the call. The Assembly Language calling sequence is:

```
      EXT EXEC,PRTN
      JSB PRTN
      DEF *+2
      DEF IPRAM
      JSB EXEC
      DEF *+2
      DEF SIX
       .
       .
       .
IPRAM BSS 5 PARAMETER BUFFER
 SIX  DEC 6 PROGRAM TERMINATION CODE
```

The FORTRAN calling sequence is:

```
DIMENSION IPRAM(5)
    .
    .
CALL PRTN(IPRAM)
CALL EXEC(6)
```

The second routine is called PRTM, passes four parameters, and does not clear the wait flag. When the parameters are recovered with RMPAR, the first parameter is meaningless. The Assembly Language calling sequence is:

```
      EXT PRTM
      JSB PRTM
      DEF *+2
      DEF IPRAM
       .
       .
       .
IPRAM BSS 4
```

INDIRECT ADDRESS SUBROUTINE

This routine is used to find an indirect address. The Assembly Language calling sequence is:

```
EXT .DRCT
JSB .DRCT
DEF ADDR
 -return-
```

The routine returns with the A-Register set to the direct address of ADDR, the B-Register unaltered, and the E-Register lost. This routine is usually used when ADDR is external.

BREAK FLAG TEST SUBROUTINE

This routine tests the break flag and if set clears it. The FORTRAN calling sequence is:

```
IF (IFBRK(IDMY)) 10,20
```

Where:

10 branch will be taken if the break flag is set. The flag will be cleared.

20 branch will be taken if the break flag is not set.

IDMY must be used in order to inform the FORTRAN compiler that an external Function is being called.

In the Assembly Language calling sequence:

```
JSB IFBRK
DEF *+1
 -return-
```

The A-Register will = -1 if the break flag is set and = 0 if not. The break flag will always be cleared if set.

FIRST WORD AVAILABLE MEMORY SUBROUTINE

This routine finds the address of the first word of available memory for a given ID segment. The Assembly Language calling sequence is:

```
EXT COR.A
LDA IDSEG
JSB COR.A
 -return-
```

The ID segment address is loaded into the A-Register and the routine is called. On return the A-Register contains the first word of available memory (MEM2 from ID). Note that on entry into a segment, the A-Register contains the segments ID segment address.

CURRENT TIME SUBROUTINE

This routine reformats and returns the time in milliseconds, seconds, minutes, hours, and the day. The FORTRAN calling sequence is:

```
CALL TMVAL (ITM,ITMAR)
```

Where:

ITM is the two word negative time in tens of milliseconds. This double word integer can be obtained from the system entry point $TIME or the time values in the ID segment.

ITMAR is a five word array to receive the time. The array is set up as:
tens of milliseconds
seconds
minutes
hours
current system day of year (not related to call values)

BUFFER CONVERSION SUBROUTINE

This routine converts a buffer of data back into its original ASCII form. The user passes the routine a buffer (IRBUF), plus the number of parameters in the buffer, that looks like the buffer returned by the PARSE routine. INPRS then reformats the buffer into an ASCII string that is syntactically equivalent (under the rules of PARSE) to a buffer that may have been passed to PARSE to form IRBUF. The length of the ASCII string in characters will be 8 times the number of parameters. The FORTRAN calling sequence is:

```
CALL INPRS (IPRBF,IPRPF(33))
```

Where:

IPRBF is the buffer IRBUF

IPRBF (33) is the number of parameters parsed

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 as 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	240_{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		
.XPAK	.PACK	3565_8	1909_{10}
.XCOM	.ZRLB		
.XFER	.OPSY		
	.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:

Resident Library = 240_{10} (Re-entrant/Privileged)

FRMTR = $+1380_{10}$ (Utility)

1 program = 1620_{10} Total FRMTR+Library

(FRMTR) = $+1380_{10}$ (Utility) 2nd Copy for 2nd Program

2 programs = 3000_{10}

(FRMTR) $+1380$ (Utility) 3rd Copy for 3rd Program

3 programs = 4380_{10} Total FRMTR Copies + Library

b. Using FORTRAN IV Formatter

Resident Library = 1940_{10} (Re-entrant/Privileged)

FMTIO = $+ 670_{10}$ (Utility)

1 program = 2610_{10} Total FMTIO + Library

$+ 670_{10}$ (Utility) 2nd Copy

2 programs = 3280_{10}

$+ 670_{10}$ (Utility) 3rd Copy

3 programs = 3950_{10} Total FMTIO Copies + Library

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

SUBROUTINE STRUCTURE

Table 4-2 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 4-2. 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 4-2. 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 4-2. 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		

PART 8
Segmented Programs

INTRODUCTION

Background disc-resident programs may be structured into a main program and several overlapping segments, as shown in Figure 4-2. The main program begins from the start of the background disc-resident area, and must be loaded during RTGEN or with the loader prior to its segments. The area for overlay segments starts immediately following the last location of the main program. The segments reside permanently on the disc, and are read in by an EXEC call when needed. Only one segment may reside in core at a time .

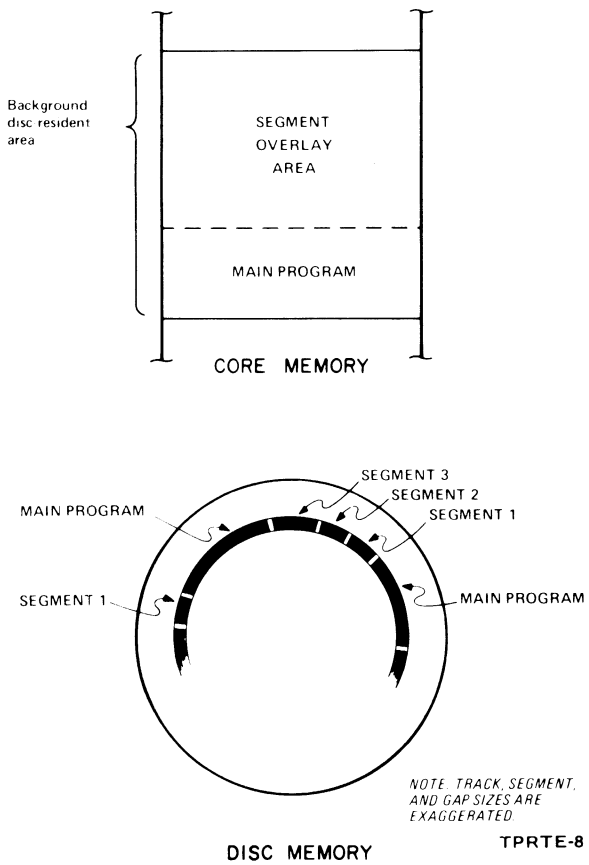


Figure 4-2. Segmented Programs

RTE ALGOL SEGMENTATION

ALGOL programs can be segmented if certain conventions are followed. The main program must be type 3, and the segment must be type 5 in the HPAL statement. The segment must be initiated using the Program Segment Load EXEC call from the main or another segment.

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

```
PROCEDURE MAIN; CODE;
```

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

RTE FORTRAN SEGMENTATION

FORTRAN programs can be segmented if certain conventions are followed. The main program must be type 3, and the segment must be type 5 in the PROGRAM statement. The segment must be initiated using the Program Segment Load EXEC call from the main or another segment.

Each segment must make a dummy call to the main program. In this way, the proper linkage is established between mains and segments:

```
      .
      .
CALL MAIN
END
```

Chaining of segments is unidirectional. Once a segment is loaded, execution transfers to it. The segment, in turn, may call another segment, but a segment written in FORTRAN

cannot easily return to the main program. All communication between the main program and segments must be through COMMON.

RTE ASSEMBLER SEGMENTATION

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

Figure 4-3 shows how an executing main program may call in any of its segments from the disc, via a "JSB EXEC." The main program is not suspended, but control is passed to the transfer point of the segment.

An executing segment may itself call in another of the main program's segments using the same "JSB EXEC" request. (See Figure 4-4). However, a segment of the FORTRAN or ALGOL compiler may not call in a segment of the Assembler.

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

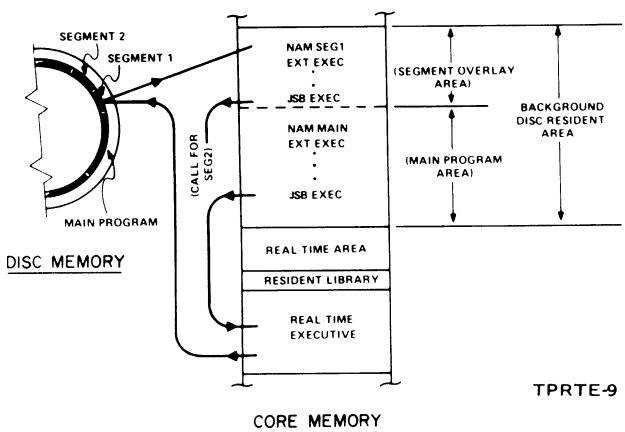


Figure 4-3. Main Calling Segment

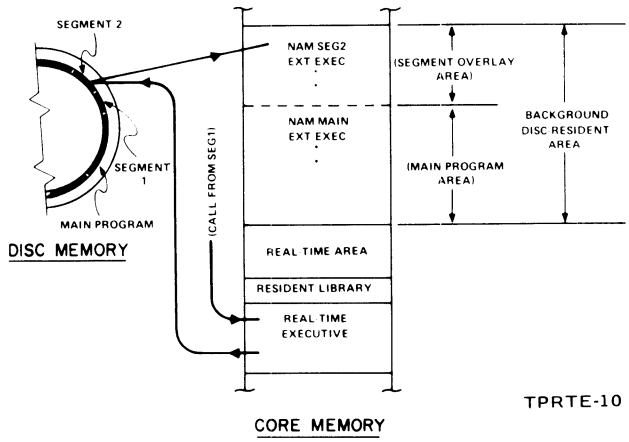


Figure 4-4. Segment Calling Segment

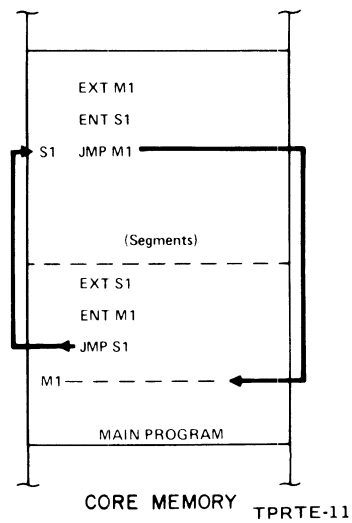


Figure 4-5. Main-to-Segment Jumps

PART 9
Multiple Terminal Operation

INTRODUCTION

Through Class Input/Output, which consists of initiating an I/O request without wait, the RTE-II System will support multiple operator consoles. This means that more than one operator can have access to the system and a single program simultaneously. To properly understand the capabilities of multiple terminal operation, one must first understand the different ways in which a single minicomputer can interact with more than one terminal.

MULTIPROGRAMMING

Under the scheme of multiprogramming, each terminal has its own unique and separate copy of a particular program. This is accomplished by renaming the program with the File Manager RN command as many times as required. When the operator at any terminal desires the system's attention he simply strikes a key which causes an interrupt. The system returns the prompt character back to the terminal signaling the operator that he has the system's attention. The operator can then activate his copy of the program until a resource limitation or higher priority interrupt occurs. For example, two users would be using the editor and one user would be using BASIC; the master copy of the editor would be on the disc, but two copies of the editor would be swapped in the background area. Most often, the operator will use the RUn command to cause a program to be initiated; for example, RU,EDITR. The multiterminal monitor will then cause the editor program to be executed using the given terminal as an input/output device. The given terminal is used because its logical unit number is placed in the first parameter following the program's name since in the above example that parameter was left out. If the user desires to use another I/O device for input/output then its logical unit number should be supplied immeidately following the program name. For example, RU,EDITR,14 would schedule the editor program and specify logical unit 14 as the I/O device.

Programs and resources are given priorities according to their interaction with the system as follows:

Real-Time Programming	–	Highest
Interactive Programming Editor BASIC FMGR	–	Middle
Bit Manipulation ASMB LOADR FTN	–	Lowest

MULTITASKING

In multitasking multiple programs exist as in multiprogramming, but the programs must communicate data and control flags to each other in order to synchronize their efforts. For example, one program may handle data gathering while another program would handle queries for statistical analysis of the data; both programs would be coordinated by a third program to provide a consistent and simple interface to the terminal user. These multitasking functions are performed in RTE-II with the following calls.

Class I/O calls
Schedule with wait
Schedule without wait
Schedule and pass parameters
Allocate resource numbers
Logical unit lock

OPERATION

Multiple terminal operation requires that two routines, PRMPT and RPN (HP Part No. 92001-16003) be configured into the system during generation and that these routines be assigned a reasonably high priority. The Interrupt Table is set up so that an interrupt generated by a terminal schedules the PRMPT routine which in turn schedules RPN. Then, as soon as any key is struck on the terminal, PRMPT issues the prompt character back to the terminal signaling the operator that he has the system's attention. Input from the operator is processed by RPN.

Any legal operator request is valid for input (e.g., ON, or ST, etc.): however, if an ON or RU command is given and

the first parameter is not specified, RPN will default that parameter to the input terminal's logical unit number. The following examples show how the multiple terminal monitor (MTM) might be used.

Example:

A key on one of the terminals is struck. The terminal responds as follows:

14>

You desire to run the Interactive Editor EDITR.

```
14> RU,EDITR
```

Since there is no parameter (specifically a logical unit number) following EDITR, MTM takes the logical unit number of the interrupting device (14 in this example) and uses it for I/O. If a logical unit number had been provided then I/O would have taken place through that device. In the above case EDITR would respond with

SOURCE FILE?

and you would be on your way using the Interactive Editor.

Programs to be scheduled for operation from several terminals must be swappable. That is, the program must perform all I/O through the re-entrant subroutine REIO instead of EXEC calls or otherwise maintain their swappability. An additional requirement is that each terminal must access the program by using a different Program ID Segment (different program name).

NOTE

Since the Logical Source (LS) and LG areas may only be used by one program at a time, it is recommended that programs such as ASMB, FTN4, etc. should not be assigned duplicate ID segments for multiple terminal operations.

When a program is to be used by several terminals it must be accessed by a different name in each case. In the above example using the Interactive Editor, it can be renamed several times on-line with the File Manager RN command. The following series of File Manager commands demonstrates how this is accomplished.

```
:PK
:RN,EDITR,EDIT1
:RP,EDIT1
:RN,EDIT1,EDIT2
:RP,EDIT2
:RN,EDIT2,EDIT3
:RP,EDIT3
:RN,EDIT3,EDITR
```

Note that the above commands can be put in a file that will be run each time the system is booted up. This relieves the user the responsibility of renaming all programs for MTM use if the system should go down and have to be rebooted.

The Pack (PK) command in the above example is issued first to recover disc space. However, if a program file is assigned to an ID segment, the disc may not be packed (see the Batch and Spool Manual).

The last rename command restores the file's original name for future use. It is recommended that a different number be assigned to each copy of the program so that the operator of each terminal may run the program without confusion as to which one is already being run by another terminal. Output which has been buffered up for a terminal may be stopped and completely eliminated by entering the Flush (FL) command.

SYSTEM CONFIGURATION

The routines PRMPT and RPN are loaded into the system during the Program Input Phase, and are assigned a reasonably high priority during the Parameter Input Phase. When the Interrupt Table is formed, and entry for PRMPT is made as follows:

select code, PROG, PRMPT

After the RTE-II System is initialized and running, each terminal must be initialized with a control request through either a File Manager command or an EXEC request.

:CN, *lu*, 20B –or– CALL EXEC (3,20B)

SECTION V
REAL-TIME INPUT/OUTPUT

INTRODUCTION

In the Real-Time Executive System, centralized control and logical referencng 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-II 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-II 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-II (these entries are established by the user when the RTE-II System is generated). These 15-word EQT entries reside in the system, and have format as shown in Table 5-1.

Table 5-1. Equipment Table Entries

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	P	S	T	Unit #						Channel				
5	AV		EQ 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															

Where:

D = 1 if DMA required.

B = 1 if automatic output buffering used.

P = 1 if driver is to process power fail.

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.

EQ TYPE CODE = type of device. When this octal number is linked with "DV*r*," it identifies the device's software driver routine as follows:

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 device.
10 = plotter.
12 = line printer.
15 = mark sense card reader.
20 to 37 = magnetic tape/mass storage devices.
30 = fixed head disc or drum.
31 = 7900 moving head disc.
32 = 7905 moving head disc.
40 to 77 = instruments.

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

When RTE-II initiates or continues an I/O operation, it places the addresses of the EQT entry for the device into the base page communication area (see Appendix A) before calling the driver routine.

DEVICE REFERENCE TABLE

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 RTGEN, and can be modified by the LU operator request (see Figure 5-1).

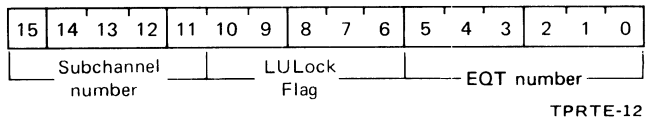


Figure 5-1. Device Reference Table

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 0 through 6 are predefined in the RTE-II System as:

0 – bit bucket
1 – system teleprinter
2 – system disc
3 – auxiliary disc
4 – standard punch unit
5 – standard input unit
6 – standard list unit

Logical units 7 through 63 may be assigned for any functions desired although logical unit 8 is recommended to be the magnetic tape device. The operator can assign EQT numbers and subchannel numbers within the EQT entries to the logical unit numbers when the RTE-II 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-II prints this message:

ILL INT *xx*

where *xx* is the octal I/O channel number. RTE-II 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-II halts unless the power fail routine is used. 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

STANDARD I/O CALLS

A user program makes an EXEC call to initiate I/O transfers. If the device is not buffered, or in the case of input transfers, the calling user program is suspended until the transmission is completed. (See Class I/O, Section III for exceptions). 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.

If the device is down, the calling program is automatically suspended in the general wait list (status = 3). While in this list the program is swappable, and if any device is set up the program is automatically rescheduled. Refer to the ST command in Section II for more information on the general wait list.

Interrupts from the device cause 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.

POWER FAIL

The system power fail routine, if loaded at generation, will perform the following steps.

a. When power comes on, will restart the real-time clock, set up a time-out entry (TO) back to its EQT, and then return to the power fail interrupt location.

b. When the EQT entry times-out, the power fail routine will check EQT word 5 bit 14 and 15 of each device. The status of bits 14 and 15 will indicate whether the device is "down" or "busy." The routine will also check bit 13 of EQT word 4 (set by driver) which indicates if the driver is to process the power fail.

c. If the device was busy when the power failed and the power fail bit is set when power resumes, the driver is entered at I.*nn* and the EQT is not reset. If the power fail bit is not set the device is set "down." The system then sets the device "up," resets the EQT and enters the driver at I.*nn*.

In other words, if the device was inputting or outputting data when the power failure occurred and the driver is written to handle power fail, when power resumes the device driver will do the power fail recovery. If the device was busy when power failure occurred and the device driver is not written to handle power failure, the routine attempts to restart the I/O operation.

d. If the device was down when the power failed and the power fail bit is set or not set, when power resumes the system "ups" the device, resets the EQT and enters the driver at I.*nn*. In other words, if the device was down when power failed, when power resumes the system "ups" the device and attempts to start the operation, if any, in the device queue.

e. An HP supplied program called AUTOR will be scheduled. AUTOR will send the time of power failure to all teletypes on the system (which re-enables all terminals). AUTOR is written in FORTRAN, with the source tape supplied so the user can easily modify the program to suit his individual needs.

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-II, 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-II operation makes it mandatory that a synchronous device driver use a DMA or privileged interrupt channel for data transfer.

Many of the I/O drivers are documented in the RTE-II Device Subroutine Library manual, HP Part No. 29100-93007.

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*xnn* and C*xnn* are the entry point names of the two sections respectively, and DV*ynn* is the driver name. As shown, the driver name is five characters long and starts with the characters "DV" and ends with a two-digit octal number (e.g., DVR00). This name is usually obtained from the software box that the tape is located in. The entry point names are four characters in length and start with either "I" or "C" and usually end with the same two-digit octal number used in the driver name. However, since the system generator RTGEN does not examine the driver's NAM record, the driver may in fact be renamed to support more than one device. The rules for the choice of "*x*" and "*y*" above are as follows:

If "*y*" is not "R" then "*x*" = "*y*"

If "*y*" is "R" then "*x*" = "."

Using the above rules, a driver named DVR16 has entry points named I.16 and C.16. A driver named DVP16 has entry points IP16 and CP16. This allows one driver to support more than one device type.

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-II calls the 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-II 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 10 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 5-2).

a. Checks for power fail entry by examining bit 15 (=1) of EQT word 5. This bit is set only on power fail entry (see "b" in Power Fail).

b. Rejects the request and proceeds to "g" if:

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

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

d. Initializes DMA, if appropriate.

e. Initializes software flags and activates the device. All variable information pertinent to the transmission must be saved in the EQT entry because the driver may be called for another device before the first operation is complete.

f. Optionally set the device time out clock (EQT 15).

g. 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.
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.)
If A = 5,	DMA channel required.

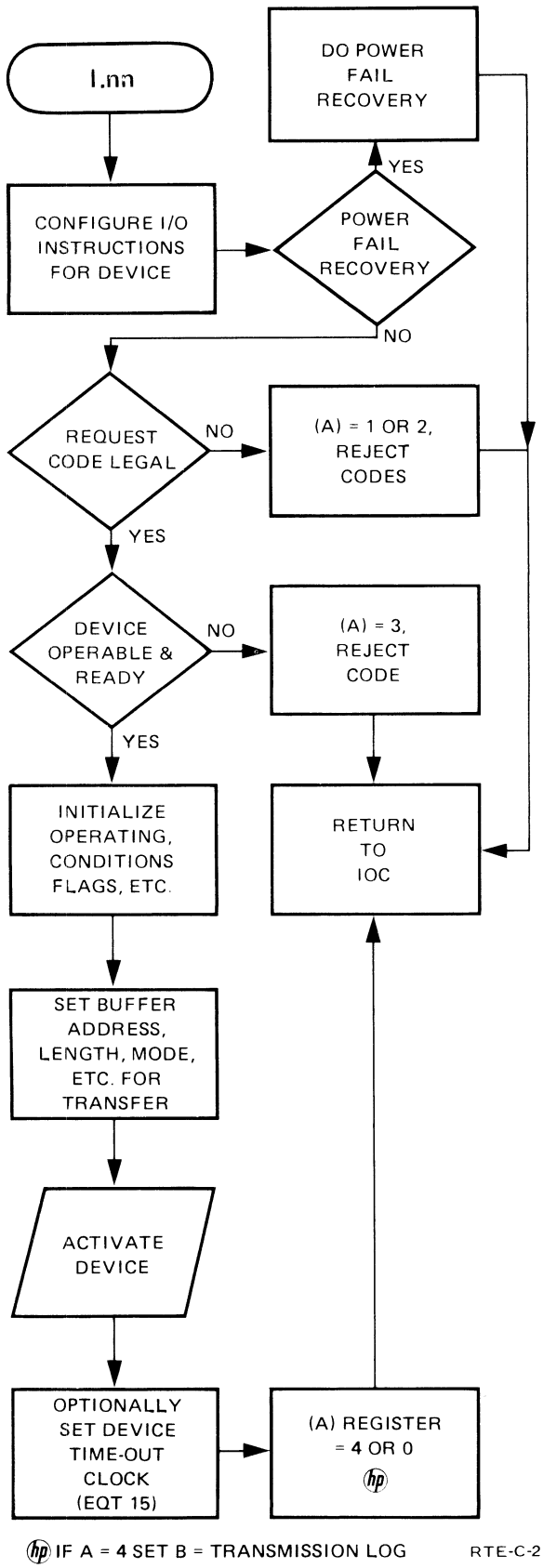


Figure 5-2. I/O Driver Initiation Section

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

a. The channel can be assigned during generation by entering a "D" in the driver's Equipment Table Entry.
b. 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 communication area. When bit 15 is set, the rest of the word must not be altered. The operation can be performed only if DUMMY is non-zero (meaning the system includes privileged interrupt processing.)

The following code demonstrates these principles:

```
CLF 0                    Disable interrupts.

STC DMA,C                Initiate DMA.

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

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

LDA B,I                  Set bit 15 of the entry equal
IOR = B]00000            to 1 and return to the inter-
STA B,I                  rupt table. Enable interrupt
STF 0                    system.
```

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 generation 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 generation 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-II 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 5-3 and performs the following functions in the order indicated.

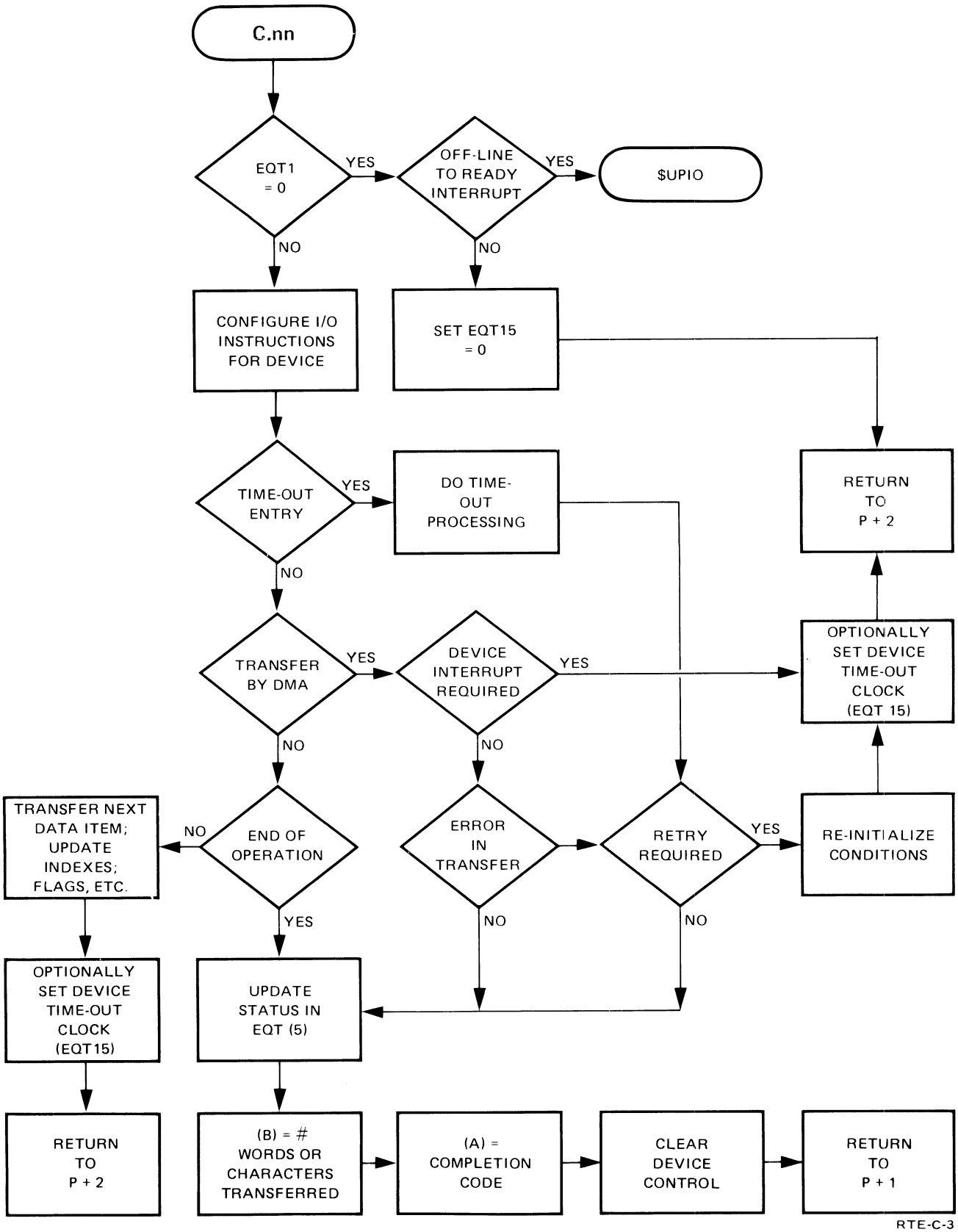


Figure 5-3. I/O Driver Completion Section

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 completion.
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 initated, is set to a value *m*, where *m*, is a negative 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 generation (see 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 except on initiation if EQT15 is not zero, it is not reset. However, a time-out value inserted directly into EQT15 by the driver over-rides any value previously set by the system (from EQT14).

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 takes 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 generation. 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.

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.

DRIVER AUTO UP

A driver has the capability of automatically "uping" itself through a JMP instruction. For example, if a driver makes a not ready, parity error, EQT, or time-out return to the system, and subsequently detects an interrupt (or time-out) entry which signals that the device is now ready, it may "up" itself as follows:

```
JMP  $UPIO
```

If any requests are pending the system will call the driver at I.*xx*.

SAMPLE I/O DRIVER

The sample driver in Figure 5-4 demonstrates the principles involved in writing an I/O driver for the RTE-II System. Note that this driver is for tutorial purposes only and not one of the drivers supplied with the system.

```
PAGE 0002 #01  ** 81 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.
```

Figure 5-4. Sample I/O Driver

```
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> S.P.K. (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.
```

Figure 5-4. Sample I/O Driver (continued)

```
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*
```

Figure 5-4. Sample I/O Driver (continued)

```
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*
```

Figure 5-4. Sample I/O Driver (continued)

```
 PAGE 0006 #01  < DRIVER /0 *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*
```

Figure 5-4. Sample I/O Driver (continued)

```
 PAGE 0007 #01  < DRIVER /W *COMPLETION SECTION* >


0178*
0179* CONSTANT AND STORAGE AREA
0180*
0181  000000          A      EQU 0           A-REGISTER
0182  000001          B      EQU 1           B-REGISTER
0183*
0184  000103 102500  LIA    LIA 0
0185  000104 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*
      000105 000003
      000106 000001
      000107 000002
      000110 000004
      000111 000077
      000112 177400
      000113 000100
      000114 001100
      000115 004000
0210                        END
**  NO ERRORS*
```

Figure 5-4. Sample I/O Driver (continued)

PRIVILEGED INTERRUPT PROCESSING

When a special I/O interface card, HP 12620, is included in the system, RTE-II allows a class of privileged interrupts to be processed independently of regular RTE 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. 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-II 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-II is running, and they can interrupt any RTE operation.

PRIVILEGED INTERRUPTS

The privileged interrupts can be recognized within 110 * n machine cycles where n is the relative priority of the privileged device's I/O card. For example:

select code 10 – n = 1
select code 11 – n = 2
select code 12 – n = 3

The privileged interrupts are 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-II.

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-II 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-II 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-II 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-II, 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 Ap-

pendix 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 (see Figure 5-5), 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-II 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 in Figure 5-5. 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 stroage 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 -1 is set in EQT word 15, 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 re-schedules 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 IOM          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
```

Figure 5-5. Sample Privileged I/O Driver

```
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
```

Figure 5-5. Sample Privileged I/O Driver (continued)

```
0119            INB                     IF
0120            LDA 5,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  EOSV      OCT 0
0173  MPFSV     OCT 0
0174  EQ4       NOP
0175  EQ15      NOP
0176  BIT12     OCT 10000
0177  *
0178  *
```

Figure 5-5. Sample Privileged I/O Driver (continued)

```
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
```

Figure 5-5. Sample Privileged I/O Driver (continued)

SECTION VI
RTE-II SYSTEM INSTALLATION

INTRODUCTION

The RTE-II system is initially configured using the system generation program, RTGEN. Configuration is accomplished by the user supplying information in response to questions from RTGEN. This section describes the steps necessary to determine the appropriate response. The section is divided into three parts.

- Part 1 contains directions for planning or laying out an RTE-II System. General directions are provided for completing the RTE-II System Configuration Worksheet, and then transferring the plans into instructions to RTGEN. If the user is quite familiar with the generation process, a punched tape containing all the parameter inputs can be made up from the configuration worksheet, and then placed in the tape reader on the system teleprinter. Each time RTGEN 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 generation 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. It is also recommended that the user read Appendix B and Section 1 at least twice before attempting to configure a system.

- Part 2 integrates the plans from Part 1 into instructions required to configure an RTE-II System based on a moving head disc drive.

- Part 3 integrates the plans from Part 1 into instructions required to configure an RTE-II System based on a fixed head disc drive.

Located at the end of Part 1 are several blank worksheets that are filled out as the system is planned. It is recommended that the worksheets be duplicated. The copies can then be used for planning purposes which leaves the original pages intact for future use. Appendix C contains the teletype/system dialog. This is an actual system configuration example intended to show the sequence of events to be expected in using the following planning process.

In this section the following terms are used:

TRACK – A software addressable (logical) track.

RTE-II SYSTEM TRACKS – The disc tracks for which RTE-II maintains a contiguous track assignment table. These tracks are located on logical unit 2 (system), and 3 (auxiliary).

PLATTER – The recording surface where the data is stored (both sides of a physical disc).

SYSTEM DISC – The disc assigned to logical unit 2. Also the platter where the absolute binary code of RTE-II resides.

AUXILIARY DISC – The disc assigned to logical unit 3. (Has same status as logical unit 2.)

SCRATCH AREA – A number of disc tracks used during system generation for storage of the relocatable binary code of RTE-II.

PERIPHERAL DISC – A peripheral disc is available to the RTE-II user for read/write operations, however, RTE-II does not manage the tracks nor maintain a track assignment table for them. A peripheral disc must have a logical unit number assignment greater than six.

PART 1
Instructions for Planning RTE-II

INPUT/OUTPUT PLANNING

I/O interface cards for peripheral devices are assigned priorities and logical unit numbers. Tables are built that affect communication between the devices and the 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 slot). The octal select codes start at octal 10 and continue up to octal 77, limited by the I/O capacity of the particular computer and any attached extenders.

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. Devices requiring privileged interrupt are always assigned to the highest priority addresses, while direct memory access (DMA) devices are assigned the lowest. The one exception to the DMA rule is in regard to the moving head system disc controller. For the fastest interrupt response, assign moving head disc controller to the next available I/O slots after the Time Base Generator (TBG).

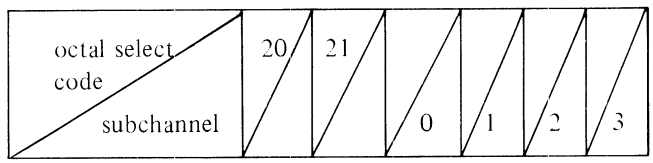
The following instructions are keyed to the I/O Configuration Worksheet in Table 6-1. Fill in the blanks as you plan your system.

STEP 1: I/O LOCATIONS

Considering the factors given in the preceding paragraphs and the instructions given below, select the priority addresses 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 two HP 7900 moving head disc drives (four subchannels) are connected to a controller in select codes 20 and 21, the top portion of the table would be completed as follows:



This method of noting subchannel numbers will facilitate assigning logical unit numbers later in the Device Reference table. For applicable HP7905 disc drive subchannel numbers, see the HP 7905 Disc Worksheet, Table 6-7.

The following detailed steps show how to assign select codes to devices starting at the highest priority address, octal select code 10. In addition to these steps, make certain that any peripheral devices or subsystems that use multiple 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 response time requirements (i.e., time from interrupt to service).

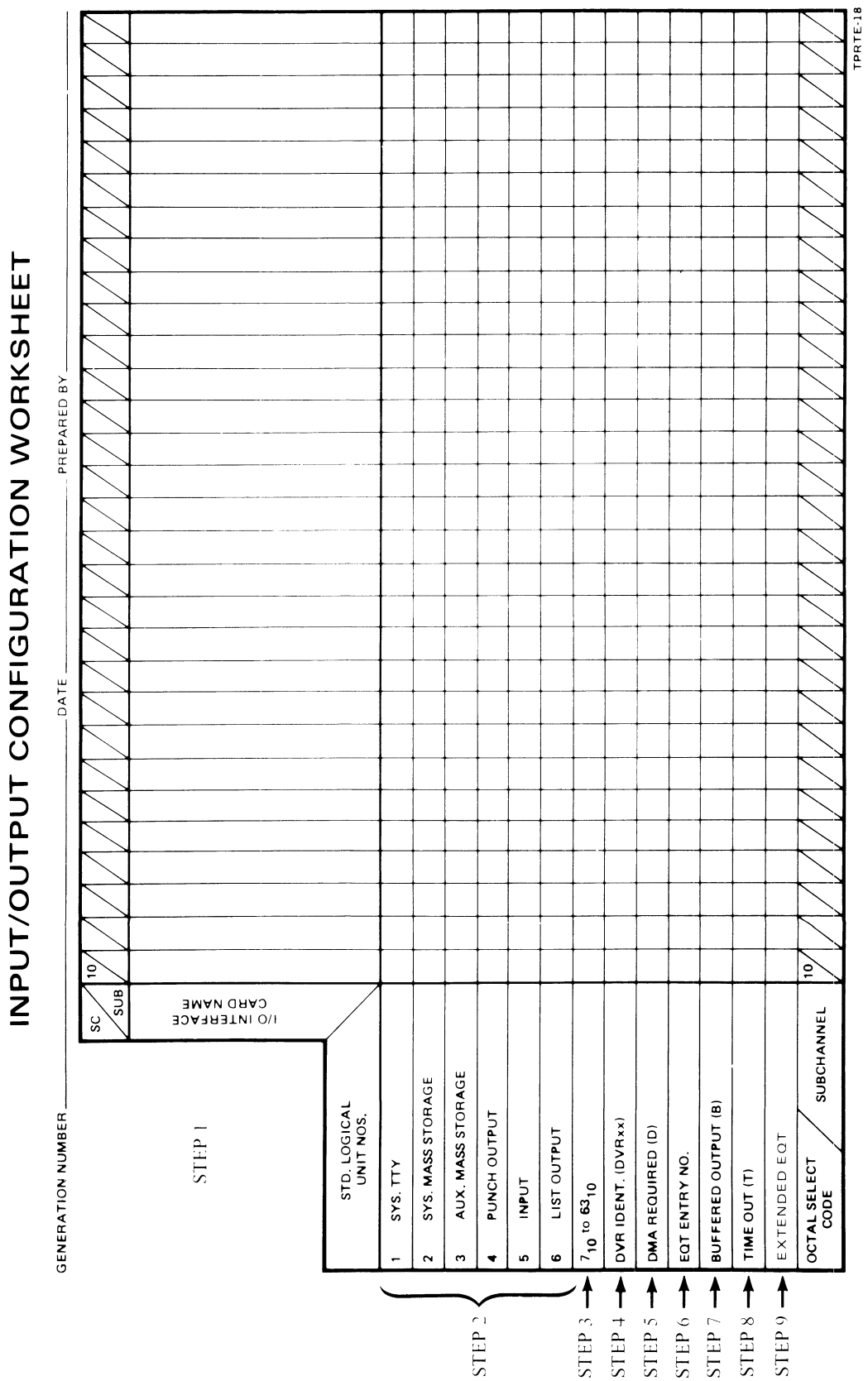
b. After the privileged devices, assign the privileged interrupt I/O card HP 12620.

c. Assign the TBG I/O card HP 12539.

d. Assign the moving head disc controller I/O cards if a moving head disc is to be configured into the system.

e. Assign all devices that *do not* use DMA in order of decreasing speed.

Table 6-1. I/O Configuration Worksheet



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 location. 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 of equivalent or even slightly slower interrupt rates.

f. Assign all devices that *do* use DMA in order of decreasing speed.

g. If an I/O extender is required and the extender does not have DMA capability, the order of steps "e" and "f"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. Place an X under one of the subchannels for LU2; include LU3 if applicable. Any remaining disc subchannels can be assigned logical unit numbers greater than six (i.e., they become peripheral if desired).

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. However, if a magnetic tape unit is being configured into the system it is recommended that it be made LU8. The power fail routine should be the last (or highest numbered) LU.

NOTE

If a device has two I/O cards use only the highest priority (lowest select code) I/O card for steps 2 and 3.

STEP 4: DRIVER IDENTIFICATION

Write in the driver identification number for each device; e.g., teleprinter driver is DVR00. If the moving head disc drive is used, in addition to placing DVR31 under the high-priority card, place a large "I" under the low-priority card. For other 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. In the case there is more than one driver with the same DVR number, refer to the paragraph under Equipment Table Entries later in this planning part.

STEP 5: DMA

Write in a large "D" for DMA required on each device that will use this capability. Note that some drivers, such as DVR62 for the HP 2313 Sub-system, are capable of dynamically assigning a DMA channel to themselves when required and do not require the D. 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. The system disc should be EQT number 1 to permit special priority assignment to an available DMA channel. Other DMA devices should then be assigned EQT numbers in order of their DMA priority. A device that has subchannels is assigned the same EQT number for each subchannel. It is recommended that whenever possible, the EQT number be the same as the LU number. This will aid the user in operating the system after it is running. It is also recommended to make the power fail routine the last (highest numbered) EQT.

STEP 7: BUFFERING

Write in a large "B" for devices that will use output buffering. Buffering means that the computer will copy into a system buffer, data that is to be output to a device (e.g., line printer).

The system will allow a program to continue processing after issuing a WRITE request to such a device, rather than suspending the program while it waits for a buffer (in the program) to be emptied.

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.

STEP 9: EXTENDED EQT

Write in a large "X" for drivers that will use the extended EQT feature. For example, each entry for the Spool Monitor Driver DVS43 will use the EQT extension. Values will be assigned later on the configuration worksheet.

DISC PLANNING

RTE-II is a disc-based operating system where a disc device provides the primary storage area for the following items:

- The configured operating system
- Relocated disc resident programs
- Relocatable library modules
- Temporary storage for programs (source for editing, relocatable output of Assembler, and so forth)
- User files

Disc tracks are grouped together to form subchannels. A scratch (work) area is set aside to temporarily hold programs during the generation process. After generation, subchannels are normally referenced through logical unit numbers which are assigned in the I/O planning section. The primary purpose of the disc planning section is to configure available disc storage into one or more subchannels. RTE-II further distinguishes between these as system, auxiliary, and peripheral subchannels. The generator will interact with the user to define a group of subchannels on a single disc controller. Multiple controllers, and mixed disc types are discussed here under the heading "Extra Disc Controllers" and in Appendix B.

SYSTEM/AUXILIARY SUBCHANNELS

The RTE-II system disc tracks are those for which RTE-II controls and maintains a track usage table. Programs may obtain tracks from, and release tracks to, this area using calls to the executive. System tracks include all tracks on the system subchannel (LU2) and the optional auxiliary subchannel (LU3). The system disc tracks are used for swapping, and by the editor, Assembler, and compilers for source, LG track, and scratch areas. In addition, system disc tracks may be used by user programs for storage.

The difference between a system and an auxiliary subchannel is that the configured system (including the memory resident system, the relocated disc resident programs, and the relocatable library) is stored on the system subchannel.

The size of a system or auxiliary subchannel is limited to 256 tracks. This number may be further reduced depending on the type of disc used (for example, 203 tracks for an HP 7900 disc).

NOTE

More than one system or type of system can be located on a disc, and those systems may share tracks. In designating tracks, those that are shared would be included and declared during each system generation. The restriction is that any tracks of an RTE-II system that are assigned to LU2 or LU3 must be unique to that RTE-II system. Remaining tracks on the disc can be assigned to other systems.

PERIPHERAL SUBCHANNELS

Disc subchannels other than system and auxiliary are classified as peripheral and must be assigned logical unit numbers greater than 6. Tracks on these peripheral subchannels are not subject to the RTE-II assignment and release mechanism. Management of these areas may be accomplished directly by user supplied programs or by the File Manager. Peripheral subchannels to be used by the File Manager must be defined with no more than 1024 tracks.

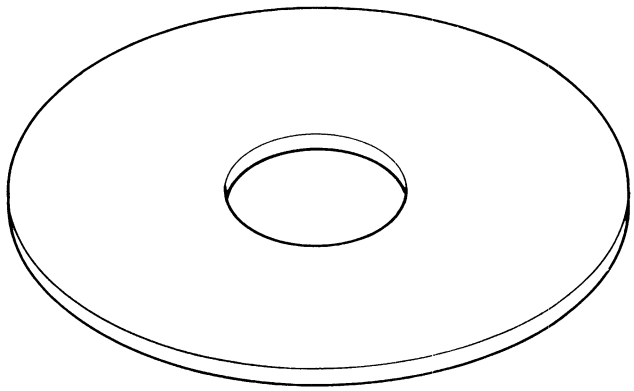
HP 7900 DISC CONFIGURATION

The HP 7900 Disc Drive is a single unit that contains two disc platters; one permanently mounted and designated subchannel 0, and the other housed in a removable cartridge and designated subchannel 1. The drive is interfaced to the computer through a plug-in controller occupying two I/O slots. It is possible to link up to four drives to the same controller which provides up to eight disc platters. Each disc platter is a subchannel, and is accessed through a logical unit number that is referenced back to the equipment table (EQT) entry number of the controller. That is, one controller, servicing eight subchannels linked to eight logical unit numbers, can control up to eight disc platters. Refer to Table 6-2 and fill in the blanks according to the following instructions.

Table 6-2. HP 7900/7901 Moving Head Disc Worksheet

SUBCHANNEL 1

REMOVABLE



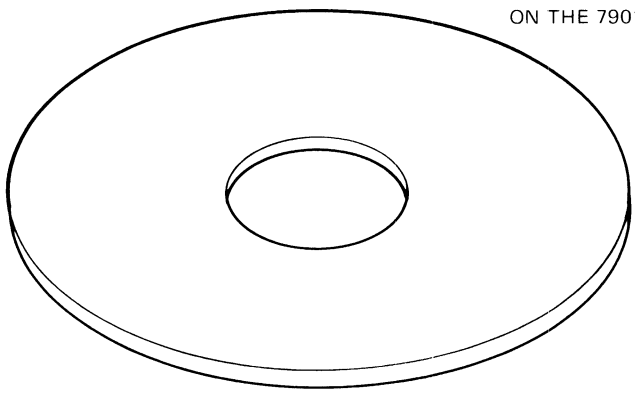
NO. OF TRACKS AVAILABLE _____

FIRST TRACK _____

NOTE:
THE FIXED PLATTER
DOES NOT EXIST
ON THE 7901.

SUBCHANNEL 0

FIXED



NO. OF TRACKS AVAILABLE _____

FIRST TRACK _____

SYSTEM SUBCHANNEL NUMBER _____ (LOGICAL UNIT 2)

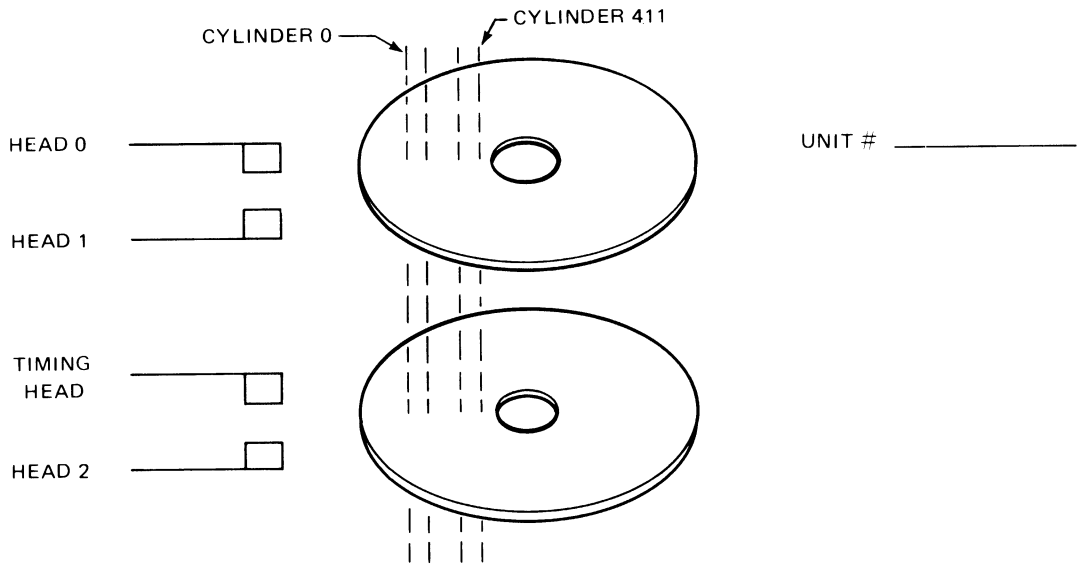
AUXILIARY SUBCHANNEL NUMBER _____ (LOGICAL UNIT 3)

SCRATCH SUBCHANNEL NUMBER _____

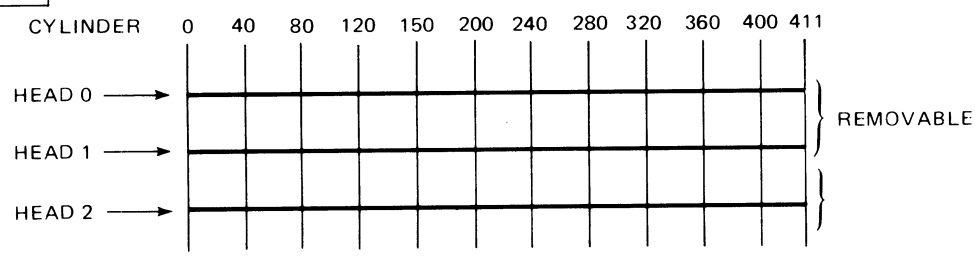
START SCRATCH (I.E. 1ST TRACK = 0) _____

Table 6-3. HP 7905 Disc Worksheet

STEP 1 **FILL IN UNIT NUMBER:**



STEP 2 **TRACKS SHOWN END-TO-END ON THREE SURFACES—CIRCLE SUBCHANNELS:**



STEP 3 **TRANSLATE STEP 2 TO NUMBERS:**

SUBCHANNEL						
NUMBER OF TRACKS						
STARTING CYLINDER						
STARTING HEAD						
NUMBER OF SURFACES						
NUMBER OF SPARES						
SYSTEM ? (✓)						
AUXILIARY (✓)						
SCRATCH ? (✓)						

Determine the number of tracks available and the starting track number for each subchannel. Then, fill in the blanks on the worksheet. Note that the maximum number of tracks available per subchannel for the HP 7900/7901 disc is 203. The moving-head Basic Binary Disc Loader (BBDL) will bootstrap a system on an HP 7900 disc only if it starts at physical track 0 on subchannel 0 or 1. Locating the system tracks anywhere else requires that a paper tape bootstrap program be punched during generation and used each time the system is loaded from the disc.

Determine which subchannel is the system subchannel and which is the auxiliary subchannel (the auxiliary subchannel is optional). Fill in the appropriate blanks on the worksheet.

Refer to the heading "Extra Disc Controllers," for instructions which cover special action required if the auxiliary subchannel is on a different controller than the system subchannel.

HP 7905 DISC CONFIGURATION

The HP 7905 Disc Drive is a single unit that contains two disc platters; one permanently mounted, and the other housed in a removable cartridge. Up to eight drives may be connected to a single 7905 controller. The controller is interfaced to the computer through an interface card occupying one I/O slot. Each disc has two surfaces; however, one surface of the fixed disc is used for timing purposes and is not available for data recording. Therefore, a single HP 7905 Disc Drive contains three surfaces (three heads) and 411 cylinders, giving 1,233 tracks, Refer to Table 6-3 for a pictorial diagram of the drive showing heads and cylinders.

The purpose of the following discussion is to configure each disc into subchannels. Each subchanel will consist of a contiguous group of tracks on a single drive, and one drive may contain several subchannels. Up to 32 subchannels may be defined on one 7905 controller. There is no physical relationship between a subchannel and a given disc area (as on 7900 discs); it is the user's responsibility to define these relationships.

The completed disc worksheet describes each subchannel on a drive in terms of the drive's unit number, size of the subchannel in tracks, starting head and cylinder numbers, surface organization, and number of tracks. In dividing up the HP 7905 disc tracks, note that the goal is a logical unit number referencing a group of disc tracks.

When filling in the worksheet on Table 6-3 there are several important rules and guidelines to remember.

- Surface organization. Tracks on a subchannel must be contiguous. Head movement should be kept to a minimum for the fastest response time to sequential tracks. This means that track assignment should alternate between surfaces. For example, if track 0 (of the first subchannel) is accessed by head 0, cylinder 0, and track 1 is accessed by head 1, cylinder 0, physical head movement (changing cylinders) is kept to a minimum.

  NOTE

  If a subchannel involves both fixed and removable platters, some flexibility is lost because removal of one platter invalidates all data on the subchannel. Additionally, the rotational alignment between two platters depends on drive orientation when the cartridge is inserted. This makes track-to-track access time across platters unpredictable. It may, in fact, be better or worse than on one platter depending on alignment and the time required for software processing between tracks.

  If more than one surface is to be used, tracks are cyclically allocated downward and back to the starting head assignment when necessary. For example, a subchannel beginning with head 1 and using 2 surfaces will use head 1 and 2 repeatedly, and in that order.

- Spare tracks. Some tracks on a disc surface may be unusable. When an unusable track is encouuntered, another track is assigned by RTGEN in its place, and the disc controller will automatically switch to that track on future references. During generation, spare tracks are assigned to each subchannel for this purpose; then, when a bad track is encountered, a subchannel may draw from its spares. Note that spare tracks are allocated on a subchannel basis and belong to that subchannel. That is, one subchannel cannot take spare tracks from another subchannel. The user should plan on about 1200 usable tracks per drive, dividing the remaining 33 tracks as spares among the subchannels in proportion to subchannel size. Spares immediately follow the main tracks for the associated subchannel, and use the same surface organization. Spares are recommended even though they may not be used on a given disc. Without sufficient spares, a subchannel or complete disc might later be copied to another disc where bad tracks are encountered, in which case all data would not fit on the new disc.

- Subchannel size. A subchannel to be used as the system or auxiliary subchannel (LU2 or 3) must not exceed 256 tracks, excluding spares. Similarly, a peripheral subchannel to be used by the file manager must not exceed 1024 tracks, again, excluding spares. Larger subchannels may be defined for access by user-developed programs.

NOTE

If the user plans to run disc utility programs designed for a 7900 disc, subchannels should be restricted to 203 tracks or less.

- Subchannel numbering. Subchannels on a given disc controller are numbered sequentially from 0. Do not skip or duplicate any numbers, otherwise the disc addressing scheme is completely up to the user.

- System subchannel. The moving head Basic Binary Disc Loader will load a system on a 7905 disc only if it starts a cylinder 0, head 0, 1, or 2. Locating the system subchannel anywhere else will require that a paper tape bootstrap be punched during generation and used each time the system is initialized.

With the aid of Table 6-3, 7905 subchannels are defined in a manner directly translatable for input to the generator. Copies of the table have been completed for two sample one-drive systems and are included as suggested disc configurations in Appendix C.

Follow the instructions below for each HP 7905 drive.

STEP 1 – A hardware unit number is associated with each drive and is selected by a switch located behind the perforated front panel. Write the appropriate unit number on the worksheet.

STEP 2 – The second part of the worksheet represents the three surfaces of the disc drive and is provided as an aid in dividing up the surfaces into subchannels. Using Table C-1 as an example, allocate to subchannel 0, 256 tracks for data and 8 tracks for spares encompassing 2 surfaces. This makes a total of 264 tracks or 132 cylinders. Note that the example in Table C-1 has all of the tracks for subchannel 0 enclosed and labeled. The first cylinder contains the first and second addressable track:

first track = head 0, cylinder 0
second track = head 1, cylinder 0

Divide up the surfaces, grouping the tracks into subchannels. Allow approximately six spare tracks for each 200 data tracks allocated. The number for the first cylinder of succeeding subchannels is found by adding the number of cylinders used by preceding subchannels. (Add tracks and spares, then divide by the number of surfaces to count cylinders.) In the example above, 132 cylinders were assigned to subchannel 0 (256 tracks plus 8 spares). Therefore, the "First Cyl" for subchannel 1 would be cylinder 132, head 0 or 1, or cylinder 0, head 2. It depends on how you assign the tracks.

STEP 3 – The third part of the worksheet answers all the questions the generator will ask about each subchannel. For the most part, the numbers are filled in from Step 2. Refer to Table C-1 for the example. Fill in the blanks for all subchannels created in Step 2.

Determine which subchannel will be the system and which subchannel will be the auxiliary (if any) and check the appropriate boxes.

EXTRA DISC CONTROLLERS

The RTE-II generator assumes a single disc controller for purposes of interactively defining and initializing subchannels. If a system is to have more than one controller (same or different disc types), the user must construct a table, according to the directions in Appendix B, describing the subchannels of the controller before beginning generation. The generator will not initialize these subchannels. The user must include the appropriate disc driver and define an equipment table entry and logical unit numbers for the subchannels (described in I/O configuration planning).

The optional auxiliary subchannel may be placed on a different controller than the system subchannel. The preceding discussion applies in this case with the added requirement that the user specify the number of tracks in the subchannel when the generator inquires about the auxiliary option (see part 2 of this section).

FIXED HEAD SYSTEM DISC

Unless otherwise stated, the following discussion deals with logical disc tracks. The number of tracks available on the system and auxiliary discs (LU2 and LU3) depends on two factors, model number of the disc drive, and if any of the disc tracks are already being used. Determine the number of available tracks by referring to Table 6-4, Fixed Head Disc Worksheet, and subtracting any tracks allocated to another software system.

The number of hardware protected tracks is selected by removing diodes on the HP 12606 or 12610 Disc/Drum

Controller cards. Refer to the individual I/O card manuals to determine if the diodes apply to logical or physical tracks. (Four physical tracks = one logical track.) The number of software protected tracks is selected by the user and entered through the teleprinter during RTGEN. The average RTE-II System requires approximately six to eight tracks for the absolute code (see the sample RTGEN in Appendix C). Compare the RTGEN listing in Appendix C to the list of software intended for this generation to estimate the approximate number of tracks to protect. For example, if the system will use more than eight logical tracks, but less than 16, the user has the option of hardware protecting 16 tracks, or hardware protecting only eight, but software protecting a greater number. Always remember, tracks for the absolute code of other software operating systems should also be protected and contiguous with the RTE-II tracks within the hardware protected area. Enter the number of the worksheet.

CAUTION

Never hardware protect more tracks than will be declared as software protected.

Table 6-4 Fixed Head Disc Worksheet

FIXED HEAD SYSTEM DISC

DRUM DISC MEMORY CHARACTERISTICS

DRUM	DISC	* LOGICAL TRACKS	SECTORS PER TRACK
–	2766	32	128
2773	–	48	
2773-003	2766-002	64	
2774	2766-003	96	
2774-003	2766-004	128	
–	2770	32	90
–	2770-01	64	
–	2771	64	
–	2771-01	128	

DIODES 12606 I/O	* LOGICAL PROT. TRKS	DIODES 12610 I/O
NONE	1	CR1-2
CR1	2	CR1-3
CR1-2	4	CR1-4
CR1-3	8	CR1-5
CR1-4	16	CR1-6
CR1-5	32	CR1-7
CR1-6	64	CR1-8
CR1-7	128	CR1-9
–	192	CR1-10

SYSTEM (LU2)

AUXILIARY (LU3)

NO. OF TRACKS AVAILABLE _____ NO. OF TRACKS AVAILABLE _____

START SCRATCH _____ SECTORS TRACK _____

NO. OF PROTECTED TRACKS _____

SECTORS TRACK _____

SYSTEM SUBCHANNEL NUMBER _____

AUXILIARY SUBCHANNEL NUMBER _____

SCRATCH SUBCHANNEL NUMBER _____

START SCRATCH (I.E., 1ST TRACK = 0) _____

Determine the number of 64 word sectors per track from the table on the worksheet and fill in the blanks on the worksheet.

GENERATOR SCRATCH AREA

RTGEN requires a scratch area for storing relocatable modules used to build the system. This area is defined only for the duration of the generation and may be placed on any of the subchannels. Two factors must be considered in selecting the size of the scratch area:

a. The area must be large enough to accommodate all of the relocatable modules, otherwise, an ERR17 will occur (see RTGEN Error Messages in this section).

b. The area must not be so large that the system area will overflow into it during the disc loading phase causing an ERR38. (See RTGEN Error Messages in this section).

An ERR38 can occur only when the scratch area is located on the system subchannel. Because the absolute system is built upwards toward the relocatable modules, it is possible that these modules may be overlayed by the system before they have been converted into absolute code. It is recommended that the scratch area not be located on the system subchannel.

NOTE

If the scratch area is assigned to a subchannel other than the system subchannel, that subchannel should not have tracks shared with another system, or any data on it that must be retained. This is because the scratch subchannel tracks assigned to the system being generated are initialized by RTGEN. As a result, any data on them is destroyed.

Determine which subchannel will provide the scratch area and indicate this on the appropriate disc worksheet. If the scratch area must be located on the system subchannel, it is recommended that the entry for the start scratch area be zero. A zero entry causes RTGEN to start the scratch area at the midpoint of the available disc area. Note that this default occurs only when the scratch area is located on the system subchannel. If either of the error codes ERR17 or ERR38 are encountered during generation, use the data in Table 6-5 as a guide to adjusting the scratch area. A formula for determining the approximate number of 64-word sectors a user-written program will occupy follows:

Number of 64-word Sectors = $\frac{x}{33}$

where, x is the number of words of main memory the program occupies.

NOTE

Table 6-5 and the formula are only approximate guides to be used as an aid if using the scratch default does not work, or if difficulty is experienced in estimating some other starting point for the scratch area.

The first logical track number of the scratch disc is always zero (0) regardless of the actual track address. For example, if the scratch is located on a subchannel consisting of cylinders 100 to 200, the starting logical track for the scratch disc would be track zero (0). To start the scratch area on a track inside the available area, count the number of tracks into the area and use that number as the starting track (e.g., to skip the first 10 tracks, start scratch on track 10).

Table 6-5. Approximate Number of 64-Word Sectors Required to Store RTE-II in Relocatable Format

NAME	64-WORD SECTORS
Executive Software	184
RTE-II System Library	50
RTE-II ASMB	174
RTE-II FORTRAN	348
RTE/DOS FORTRAN IV Compiler	464
RTE/DOS FORTRAN IV 10K Compiler	354
RTE-II ALGOL	176
RTE-II Editor	38
RTE-II Interactive Editor	66
RTE-II Loader	142
RTE/DOS Relocatable Library	240
RTE/DOS FORTRAN IV Library	290
RTE/DOS HP FORTRAN Formatter	42
RTE/DOS Plotter Library	78
Drivers	Allow 11 sectors per driver

The next worksheet to be completed depends on the type of disc to be initialized. Use either Table 6-6, HP 7900/7901 Moving Head Disc Initialization; Table 6-7, HP 7905 Moving Head Disc Initialization; or Table 6-8, Fixed Head Disc Initialization. The disc initialization worksheet contains specifications of all disc tracks belonging to the system, memory, time base generator channel, swapping options, and program input devices.

MOVING HEAD DISC INITIALIZATION

During the initialization phase, RTGEN requests information necessary to generate a track map that defines disc subchannels. Once the track map is established, RTGEN continues requesting information necessary to generate the system.

HP 7900/7901 DISC INITIALIZATION

Fill in the blanks on Table 6-6. The information requried for the following steps can be obtained from Table 6-2, the HP 7900/7901 Disc Worksheet.

STEP 1 – Write in the lower numbered/highest priority select code (I/O slot) for the disc controller.

STEP 2 – Fill in the track assignments for each subchannel. A zero (0) entered for number of tracks causes RTGEN to ignore that subchannel.

Go to STEP 3.

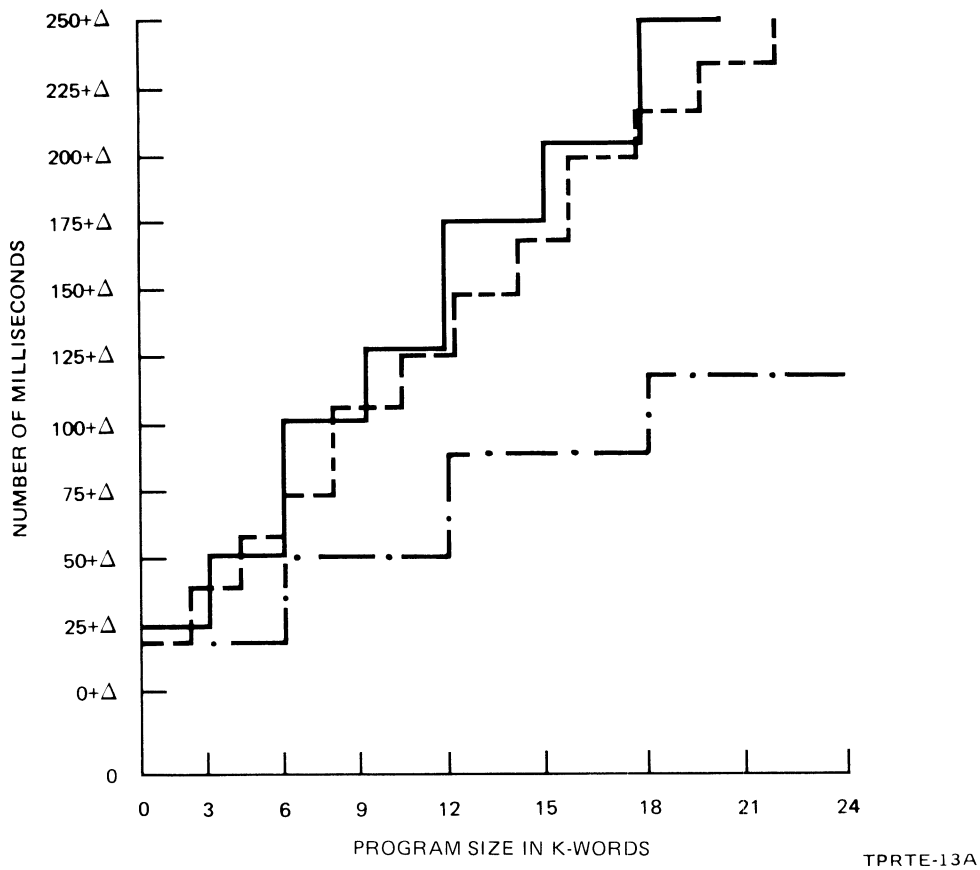
HP 7905 DISC INITIALIZATION

Fill in the blanks on Table 6-7. Most of the information required for the following steps can be obtained from Table 6-3, the HP 7905 Disc Worksheet.

STEP 1 – Write in the lower numbered/highest priority select code (I/O slot) for the disc controller.

STEP 2 – Fill in the blanks for each subchannel from Table 6-3.

STEP 3 – The number of 128-word sectors per track is 48.



THIS GRAPH REPRESENTS THE TIME IT TAKES TO READ OR WRITE A PROGRAM TO THE DISC. THE TIMES SHOULD BE DOUBLED TO GET TOTAL SWAP TIME.

NOTES:

Δ RANDOM ACCESS TIME WHICH IS COMPOSED OF TWO QUANTITIES.

A. THE ROTATIONAL DELAY. THIS DELAY RANGES FROM 0 TO ONE ROTATION TIME (RT) WITH EQUAL PROBABILITY, THUS THE AVERAGE ROTATIONAL DELAY IS RT/2.

B. THE SEEK TIME (ST). FOR A FIXED HEAD DISC ST = 0. FOR A MOVING HEAD DISC IT RANGES FROM 0 TO SOME MAXIMUM WITH A NON-EQUAL PROBABILITY. THE SEEK TIME DEPENDS ON THE LAST ACCESSED TRACK NUMBER.

——— HP 7900/7901
— — HP 2771
—·— HP 7905

EXAMPLE:

USING THE ABOVE PRINCIPLES AND GIVEN THE FOLLOWING DATA FOR AN HP 7900 DISC, WE CAN PLOT THE LOAD/SWAP TIME AS A FUNCTION OF THE NUMBER OF WORDS. NOTE THAT THE NUMBER OF WORDS IS AFFECTED BY THE "ALL OF CORE BIT."

FOR AN HP 7900: RT = 25 MS
#RT'S/TRACK = 2
#WORD/RT = 3K

Figure 6-1. Swap Delay Graph

STEP 4 – If only one subchannel is assigned to the system, place a dash (i.e., does not apply) in system subchannel, auxiliary disc subchannel, and scratch subchannel blanks. Otherwise fill in the numbers from the Disc Worksheet, Table 6-2 or Table 6-3. Refer to the pragraphs under scratch area and enter that number.

STEP 5 – Fill in the select code of the TBG card. Fill in the select code of the privileged interrupt card. If there is no privileged card, enter a zero (0).

STEP 6 – The swapping questions are answered with a YES or NO which allows or disallows foreground/background swapping. The core lock questions also are YES or NO answers. If YES is answered to either of these questions, it allows a program running in the appropriate area to lock itself into core and not be swapped. If the answer is NO then the program cannot be locked into core. Refer to the PROGRAM SWAPPING CONTROL call in Section III.

STEP 7 – The answer to the Swap Delay question is a decimal number between 0 and 255 that represents tens-of-milliseconds (i.e., 0 to 2550 milliseconds). If a number "N" is entered here a program will not be swapped if it resides in a disc resident area, is in the time list, and is to run within "N" milliseconds of the current time and has priority over its contender for that core area. The amount of time required for a program to swap depends on several factors; type of disc drive, program length, and if the program is segmented. To obtain an accurate figure tailored to core size, program size, and disc type, refer to Figure 6-1, Swap Delay Graph. Remember, the number selected here is applied to all swapable programs.

STEP 8 – Last word of available memory depends on computer memory size.

16K – 37677
24K – 57677
32K – 77677

STEP 9 – The use of input units is interchangeable. Maximum versatility can be achieved by designating one device for paper tape and one for magnetic tape if magnetic tape is present on the system. For example, if most of your relocatable programs are on magnetic tape and only a few on paper tape, it would be most efficient to set up the magnetic tape as the program input device (PRGM) and the paper tape reader as the library input device (LIBR). Note that there is no difference between programs input through the PRGM device and those input through the LIBR device. Any program may be loaded through either device. Part 2 of this section describes the mechanism for switching between devices during input.

The program input and library input devices can be:

PT – paper tape (photoreader)
TY – Teleprinter
MT – magnetic tape
DF – fixed-head disc file

The parameter input device is either PT or TY. If the answers to the generator requests are contained on paper tape, use TY. Then switch register bit 5 can be used to control the source of input (see Table 6-10).

STEP 10 – All disc tracks belonging to the system may or may not be initialized. RTGEN automatically initializes tracks on the system, auxiliary, and scratch subchannels. If other subchannels are available RTGEN asks if they are to be initialized by subchannel number in ascending order. If the answer is YES, RTGEN initializes only the system tracks on that subchannel and reports any defective tracks.

If the answer is NO, RTGEN does not initialize that subchannel.

NOTE

Any tracks on a subchannel that are shared with other systems should not be initialized because that data will be destroyed.

BAD TRACK INFORMATION

7900 DISCS – Up to 10 bad tracks are allowed before RTGEN aborts. Bad tracks in the area where the absolute system and relocatable library are stored will prevent operation of the system (this is the area reported at the end of generation).

Defective tracks are reported as shown below:

BAD TRACK SUBCHNL *x*
000*yyy*

where *yyy* is the logical track number and is needed when initializing the file manager for the subchannel *(x)* reported.

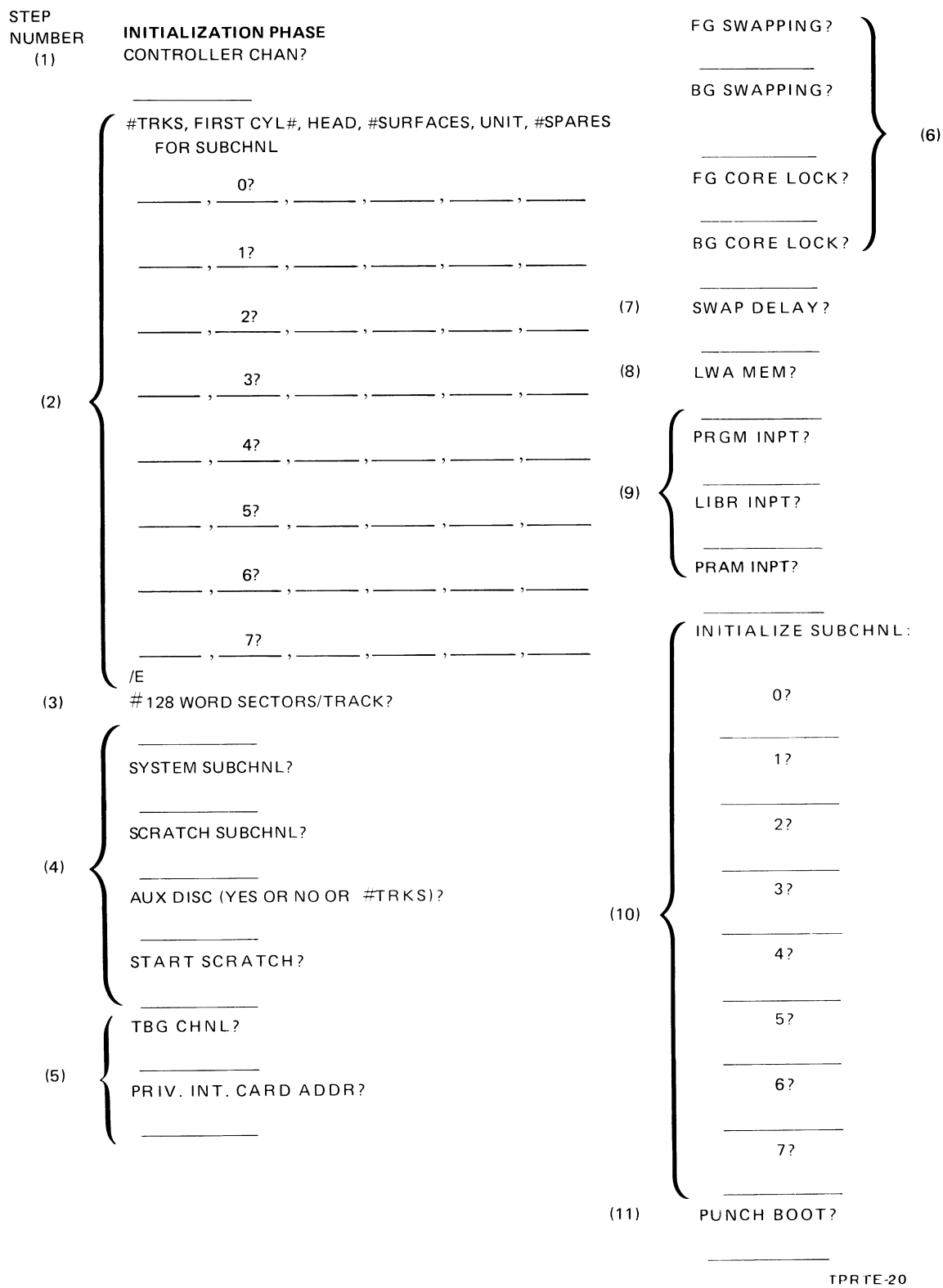
7905 DISCS – Bad tracks are automatically spared by the generator to tracks set aside for that purpose in the initialization phase. Bad tracks reported and spared during generation will not prevent operation of the system and should not be specified during file manager initialization of a cartridge on the subchannel.

Table 6-6. HP 7900/7901 Moving Head Disc Initialization

STEP NUMBER	INITIALIZATION PHASE
(1)	MH DISC CHNL?
	_____
(2)	#TRKS, FIRST TRK ON SUBCHNL:
	0?
	_____,_____
	1?
	_____,_____
	2?
	_____,_____
	3?
	_____,_____
	4?
	_____,_____
	5?
	_____,_____
	6?
	_____,_____
	7?
	_____,_____
	/E
(3)	#128 WORD SECTORS/TRACK?
	_____
(4)	SYSTEM SUBCHNL?
	_____
	SCRATCH SUBCHNL?
	_____
	AUX DISC (YES OR NO OR #TRKS)?
	_____
	START SCRATCH?
	_____
(5)	TBG CHNL?
	_____
	PRIV. INT. CARD ADDR?
	_____
(6)	FG SWAPPING?
	_____
	BG SWAPPING?
	_____
	FG CORE LOCK?
	_____
	BG CORE LOCK?
(7)	_____
	SWAP DELAY?
(8)	_____
	LWA MEM?
	_____
(9)	PRGM INPT?
	_____
	LIBR INPT?
	_____
	PRAM INPT?
	_____
(10)	INITIALIZE SUBCHNL:
	0?
	_____
	1?
	_____
	2?
	_____
	3?
	_____
	4?
	_____
	5?
	_____
	6?
	_____
	7?
	_____
(11)	PUNCH BOOT?
	_____

TPRTE-20

Table 6-7. HP 7905 Moving Head Disc Initialization



Defective tracks are reported as shown below:

	LOGICAL	CYL	HD	UNIT
BAD TRACK	xxxx	xxxx	x	x
SPARED TO	xxxx	xxxx	x	x

STEP 11 – The punch boot question is optional. If the system resides on subchannel 0 or 1 and starts at track 0, the tape is not required. Otherwise this blank requires a minimum of one YES answer. Write in YES for as many bootstrap tapes as you desire. Write in NO for termination.

FIXED HEAD DISC INITIALIZATION

Many of the answers to the following steps can be obtained from the Fixed Head Disc Worksheet, Table 6-4. For Table 6-8, perform the following steps:

STEP 1 – Write in the lower numbered/highest priority select code for the disc controller.

STEP 2 – Write in the system disc size, where the scratch starts, number of software protected tracks, and number of 64 word sectors per track. (Note that if the auxiliary disc is an HP 7900 the answer must still be the number of 64 word sectors per track.) If there is no auxiliary disc, write a zero (0) in the auxiliary disc size blank, and a dash (i.e., does not apply) in the # sectors/track blank.

Fill in the select code of the privileged interrupt card. If there is no privileged card, enter a zero (0).

STEP 3 – The swapping questions are answered with a YES or NO which allows or disallows foreground/background swapping. The core lock questions also are YES or NO answers. If YES is answered to either of these questions, it allows a program running in the appropriate area to lock itself into core and not be swapped. If the answer is NO then the program cannot be locked into core. Refer to the PROGRAM SWAPPING CONTROL call in Section III.

STEP 4 – The answer to the Swap Delay question is a decimal number between 0 and 255 that represents tens-of-milliseconds (i.e., 0 to 2550 milliseconds). If a number "N" is entered here, a program will not be swapped if it resides in a disc resident area, is in the time list, and is to run within "N" milliseconds of the current time and has priority over its contender for that core area. The amount of time required for a program to swap depends on several factors; type of disc drive, program length, and if the program is segmented. To obtain an accurate figure tailored to core size, program size, and disc type, refer to Figure 6-1, Swap Delay Graph. Remember, the number selected here is applied to all swappable programs.

STEP 5 – Last word of available memory depends on computer memory size.

16K – 37677
24K – 57677
32K – 77677

STEP 6 – The use of input units is interchangeable. Maximum versatility can be achieved by designating one device for paper tape and one for magnetic tape if magnetic tape is present on the system. The program input and library input devices can be:

PT –paper tape (photo reader)
TY –teleprinter
MT –magnetic tape
DF –fixed head disc file

The parameter input device is either PT or TY.

This concludes the Fixed Head Disc Initialization phase. Proceed to Table 6-8, System Configuration Worksheet.

SYSTEM CONFIGURATION

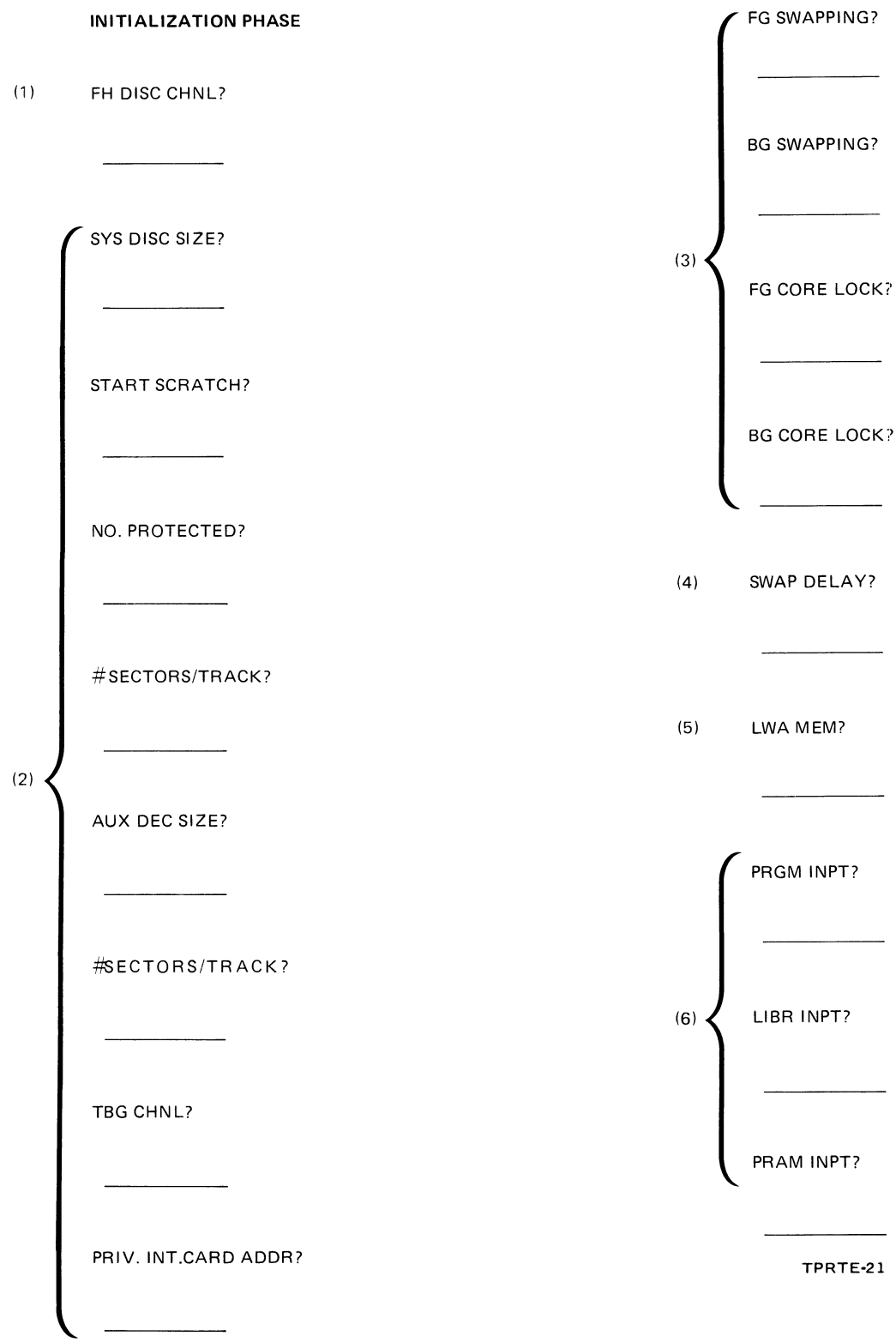
This portion of planning the generation is divided into four phases. Refer to Table 6-9, the System Configuration Worksheet. The first phase, Program Input Phase, concerns gathering the necessary tapes to be loaded and putting them in order. The second phase is the Parameter Input Phase and allows the user to modify certain parameters of the programs. The third phase is the Table Generation Phase and concerns planning all the necessary internal tables required by the system for effective communication between user programs, the system, and I/O devices. The fourth phase is the System Boundaries Phase and concerns setting up boundaries for various parts of the system, and allocating common areas.

PROGRAM INPUT PHASE

Due to the large number of tapes to be loaded during this phase, it is recommended that they be placed on a table in the following order.

Core Resident System
I/O Drivers
Power Fail (DVP43)
System Programs written by the user
Multi-Terminal Monitor
Foreground Core-Resident Programs
Foreground Disc-Resident Programs
Background Core-Resident Programs
Assembler (Main and its Segments)

Table 6-8. Fixed Head Disc Initialization



FORTRAN (Main and its Segments) and/or
FORTRAN IV (Main and its Segments) but not both FORTRAN IV Versions
ALGOL
Auto Restart
Relocating Loader
Editors
Batch Monitor
Other Background Disc-Resident Programs and their respective segments, if any.
System Library
Batch Monitor Library
Library Programs
Utility Programs

NOTE

Some of the above relocatable modules may not be present in some configurations.

PARAMETER INPUT PHASE

During the parameter input phase, the operator can modify the type, priority, or execution intervals and the ENT (entry) records of any of the programs entered during the program input phase (except that the primary type code of background main programs and their segments cannot be changed without losing their relationship to each other).

RTGEN has an additional feature that applies only to type 1, 2, 3, 4, 9, 10, 11, or 12 programs. During the Parameter Input Phase one program of this type can be scheduled to execute automatically whenever the RTE-II System is loaded from the system disc. This is accomplished by adding 80 to the program's type code. For example, if PROG is originally a type 2 program (real-time disc-resident), it can be changed to:

PROG, 82[,priority] [,execution interval]

This will cause PROG to be scheduled automatically each time the system is loaded into memory from the disc. Only one program can be assigned to type 80 code for automatic scheduling. If more than one program is assigned an 81, 82, 83, 84, 89, 90, 91, or 92 type code, only the last one entered in this phase is automatically scheduled.

Each parameter record is of this general form:

name,type [,priority] [,execution interval]

Where:

name is the name of the program.

type is the program type code:

0 – system program or driver
1 – foreground core-resident
2 – foreground disc-resident
3 – background disc-resident
4 – background core-resident
5 – background segment
6 – library, re-entrant or privileged
7 – library, utility
8 – if program is a main, it is deleted from the system

– or –

8 – if program is a subroutine, then it is used to satisfy any external references during generation. However, it is not loaded in the relocatable library area of the disc.
9 – foreground core-resident, uses background common
10 – foreground disc-resident, uses background common
11 – background disc-resident, uses foreground common
12 – background core-resident, uses foreground common
13 – background segment, uses foreground common
14 – library, core-resident

NOTE

It is illegal to classify a subroutine as type 14 that does not qualify as a type 6.

priority is the program priority from 1 to 32767 with 1 the highest priority.

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

resolution code (0 to 4):

0 – no execution interval
1 – tens of milliseconds
2 – seconds
3 – minutes
4 – hours

execution multiple (0 to 4095); the resolution code gives the units for the execution multiple.

initial absolute starting time (four values):

hours	(0 - 23)
minutes	(0 - 59)
seconds	(0 - 59)
tens of milliseconds	(0 - 99)

Fill in the blanks on the worksheet for any programs that are to be modified.

The next set of blanks are for type 3 (absolute) and 4 (replace) entry (ENT) record creating and modifying. Each ENT record takes the following form:

entry, type, value

Where

entry is the entry point name.

type is the entry point type.

AB = Absolute
RP = Replace

value is the entry point instruction value. Octal numbers are assumed unless the letter "D" follows the number which signifies decimal.

When an entry point is declared as absolute (*type* = AB) its *value* is added to the referencing instruction to obtain the final instruction value.

When an entry point is declared as replace (*type* = RP) the generator loader will replace each reference to it with the octal (or decimal) *value*. This provides the user with the capability of creating type 4 entry records which are code replacement values. This means that a JSB instruction referencing an external entry point (e.g., JSM .FAD) is intercepted by the RTE Loader and changed to a value which has been defined by the RP command. This allows the user to eliminate software subroutines by replacing their entry points with microcode instructions. For example:

.FMP,RP,105040

This would cause each JSB .FMP instruction (multiply) to be changed to floating point multiply. Other examples are shown below.

RP also allows the user to plan the system with future expansion of his programs in mind. For example, an entry point can be created with RP during generation (or later with RPL in the RTE Assembler, see that part in this manual), that is not required during generation but will be used later by programs loaded on-line.

Floating Point				*Fixed Point*		
.FAD,	RP,	105000	– Add	.MPY,RP,	100200
.FSB,	RP,	105020	– Subtract	.DIV,RP,	100400
.FMP,	RP,	105040	– Multiply	.DLD,RP,	104200
.FDV,	RP,	105060	– Divide	.DST, RP,	104400
IFIX,	RP,	105100	– Fix		
FLOAT,RP,		105120	– Float		

Another example use would be to do I/O configuration at load time, and configuring tables that are assembled as DEF statements to externals.

The next item on the worksheet concerns blank ID segments. One blank ID segment is required for each program that will be loaded permanently into the system on-line by the RTE-II relocating loader. If five segments are allocated, then only five additional programs can be loaded into the system on-line. If a temporary program is deleted from the system by an OF,*name*,8 operator command, or a permanent program is deleted from the system by the ON,LOADR,,,4 command, the program's ID segment is returned to the system to use for another on-line load. Each ID segment requires 29 words in the system core-resident area (28-word ID plus one key word). Fill in the number of blank ID segments required. (Note: 0 is changed to 1 to allow loading at least one program.)

The next item on the worksheet concerns blank background ID segments, or short ID segments. These ID segments require 10 words (9-word ID plus one key word) and are used only for background program segments. One short ID segment is required for each program segment. If an on-line load is done, and there are no blank short ID segments available, a regular 29-word one will be used.

The next item on the worksheet is for the First Word Available on Base Page for links (FWA BP LINKAGE). The first word available for base page linkages is established from the I/O Configuration Worksheet as the last used select code plus one (e.g., if the last I/O card in the priority string is located in select code 26, the FWA BP LINKAGE would be 27). Determine the number from the I/O Worksheet and fill in the blank.

TABLE GENERATION PHASE

The Table Generation Phase is the final phase that generates all the required tables, and converts the relocatable programs.

The first blank in the Table Generation Phase concerns Class Input/Output numbers. Multiple terminal operation requires one Class Number; spooling requires two, and there must be one Class Number for each Class GET call simultaneously outstanding (see Section III). For example, if you specify ten Class Numbers here, ten programs can simultaneously process class requests. Enter a number between 1 and 255 (note that 0 is changed to 1).

The next blank concerns a table (configured by the generator) called LU Mappings that cross references physical logical unit numbers to logical unit numbers within the Batch System. The number entered here is the table size and is related to the maximum number of logical unit numbers referred to in a single job within the Batch and Spool Monitor. A common entry would be ten. If the Batch and Spool Monitor is not used, zero can be entered but is defaulted to one.

The next blank concerns the allocation of Resource Numbers (RN's). Spooling requires four RN's and there must be one RN for each resource to be controlled. See the Resource Management Call in Section III. For example, if you specify ten RN numbers here, ten resources (e.g., I/O device or file) can be managed and used by cooperating programs. Enter a number between 1 and 255 (note that 0 is changed to 1).

The next blank concerns current buffer limits. Settling upper and lower memory limits here can prevent an inoperative or slow I/O device from monopolizing available system memory. Each time a buffered I/O request is made (Class I/O requests are buffered), the system adds up all the buffered words in I/O requests queued to that EQT entry and compares the number to the upper limit set here (or by the BL command). If the sum is less than the upper limit the new buffered request is added to the queue. If the sum is larger than the upper limit the requesting program is suspended in the general wait (STATUS = 3) list. When a buffered I/O request completes, the system adds up the remaining words in I/O requests queued to that EQT entry and compares the number to the lower limit set here (or by the BL command). When the sum is less than the lower limit, any programs suspended for exceeding the buffer limits on this EQT are rescheduled. An appropriate entry of 100 and 400 can be entered and later changed with the BL command if desired.

EQUIPMENT TABLE ENTRY (EQT TABLE) The first table to be completed is the Equipment Table. Each entry is located on the I/O worksheet and is to be transferred to this table. EQT number one should be the system disc and is either DVR30 for the fixed head or DVR31 for the moving head. Note that each EQT entry contains a blank for the driver name which contains five characters, starts with the characters "DV" and ends with a two-digit octal number (e.g., DV*ynn*). This name is usually obtained from the software box that the tape is located in. The entry point names are four characters in length and start with either "I" (e.g., I*xnn* for Initiation section), or "C" (C*xnn* for Completion section), and usually end with the same two-digit octal number used in the driver name. However, since RTGEN does not examine the driver's NAM record, the driver may in fact be renamed to support more than one device type. The rules for the choice of "*x*" and "*y*" above are as follows:

If "*y*" is not "R" then "*x*" = "*y*"
If "*y*" is "R" then "*x*" = "."

Using the above rules, more than one driver with the same name can be configured into the system by changing the third character in the name. For example, the system has two line printers of different types. Each line printer uses a different driver but the drivers have the same common name, DVR12. Both drivers could be configured into the system by changing the name of one to DVA12. Its entry points for the Interrupt Table would then become IA12 and CA12. The other driver would be DVR12 with entry points of I.12 and C.12. The remaining blanks on the EQT entry line are for D (DMA required), B (buffered output), T (time-out), and X (extra memory). The blanks are filled in as shown in the example in Figure 6-2.

If T is specified, a value for T must be entered in the T = blank. The value must be a positive decimal number up to 32767. This is then the number of time base generator interrupts (10 msec intervals) between the times 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 and devices controlled by drivers handling their own time-out. 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 500. Also, devices controlled by DVR00 require special subchannel assignments to make the time-out feature effective. Refer to the DVR00 Small Programs Manual, HP Part No. 29029-95001.

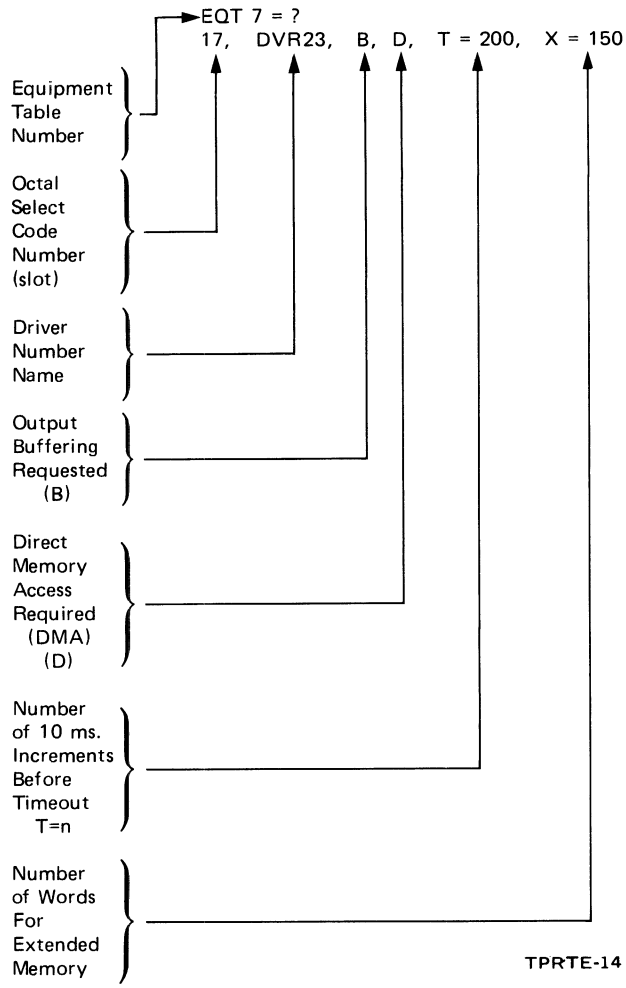


Figure 6-2. EQT Table Example

If "X" is specified (on the I/O Configuration Worksheet) a value for "X" must be entered in the "X =" blank. The value must be a positive decimal number up to three digits. This is then a number of words of buffer space allocated to the driver for its use, and is called an EQT extension. The result of this entry is recorded in the driver's EQT table, words 12 and 13. EQT word 12 contains the number of words of buffer space, and word 13 contains a pointer to the buffer. An example use of the EQT extension is for the Batch and Spool Monitor Driver DVS43. An entry must be made for each spool file that will be active, or currently doing I/O. For example, a common number of 6 files could be active at one time. The entries (referencing unused I/O slots) might be:

30, DVS43, X = 18
31, DVS43, X = 18
32, DVS43, X = 18
33, DVS43, X = 18
34, DVS43, X = 18
35, DVS43, X = 18

Refer to the I/O Worksheet (Table 6-1) and write in the octal select code number, DVR number, and D, B, T, and X options if applicable, for each EQT number in sequential order. Note that the driver's identifying suffix letter is not included.

DEVICE REFERENCE TABLE (DRT TABLE) – 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 (Table 6-1) to obtain the EQT entry number, LU number, and optional subchannel number. Fill in the blanks as shown in Figure 6-3. LU0 (bit bucket) is a system mechanism that allows immediate I/O completion. That is, the data buffer is written to or read from a non-existent device. The first seven LU numbers are reserved for system devices as follows:

LU0 – Bit bucket (no entry required)
LU1 – System teleprinter
LU2 – System mass storage
LU3 – Auxiliary mass storage
LU4 – Standard punch unit
LU5 – Standard input unit
LU6 – Standard list unit
•
•
LU8 – Recommended magnetic tape

Extra LU numbers can be assigned using EQT number zero and may be changed on-line to reference other EQT numbers as desired.

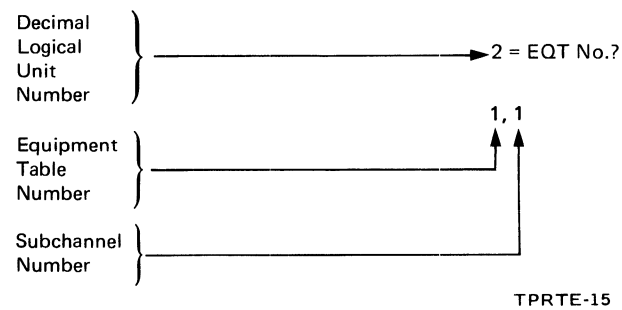


Figure 6-3. DRT Table Example

INTERRUPT TABLE (INT TABLE) – This table allows the user to establish interrupt links that tie the octal select codes back to EQT numbers. Each I/O card (select code), in ascending order, is referenced back to its EQT entry number that was established in the Equipment Table Entry part. If dummy I/O slots were used to reference EQT numbers for the Batch and Spool Monitor Driver DVS43, interrupt links for those entries are also necessary. For example (refer to the sample generation in Appendix C),

EQT number one (the first entry) was assigned to I/O slot 21, DVR31. Now, in the interrupt table, I/O slot 21 will be referenced back to EQT number one. In this manner, an interrupt occuring on I/O slot 21 will be directed to EQT number one which has the address of DVR31. The format for the entry is shown in Figure 6-4.

From Figure 6-4 the entries for the interrupt table are constructed as follows:

octal select code number is taken from the I/O Worksheet (Table 6-2) in ascending order.

option directs the system in handling the interrupt; there are four options:

select code,EQT,*n2*	relates channel to EQT entry *n2*.
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,*xxxxxx*	places an absolute octal value xxxxxx (instruction code) in the interrupt location (may be NOP, CLC, etc.)

The HP 7900 disc controller I/O cards both require an interrupt link to their EQT entry number. Reference the select code numbers to the DVR31 EQT entry number as shown below.

```
21, EQT, 1
22, EQT, 1
```

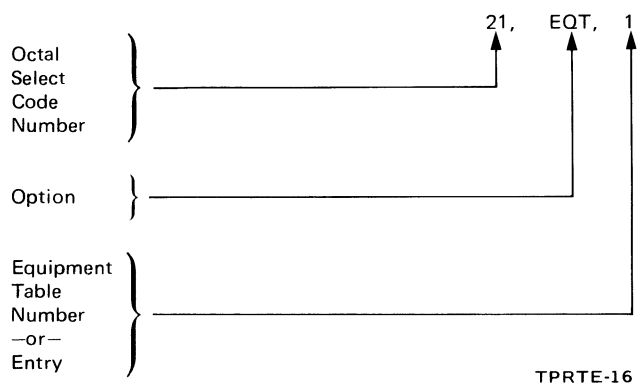


Figure 6-4. INT Table Example

For other 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, $POWR
```

where $POWR is an entry point in the power-fail routine.

SYSTEM BOUNDARIES PHASE

This ends the planning phase of RTGEN. The final questions cannot be effectively answered until RTGEN begins loading the system and reports the actual boundaries involved. Some of the boundary changes are for convenience purposes. That is, the boundary is changed to allow the affected area to begin on a page boundary, if desired. In these cases, any gaps are collected and used as system buffer memory by the system. If there is to be no change of a boundary address, enter a 0 (zero) in the change blank. Refer to 6-5 as an aid in locating the referenced areas.

The first blank is boundary "A" and concerns the library. Move the address up to start it on a page boundary, if desired.

The next blank is area "B" and is the foreground common area. The number of words allocated for foreground common is reported and then RTGEN gives the user the opportunity of increasing the area.

The next blank is boundary "C" which is the foreground core-resident address. Move the address up to start the area on a page boundary, if desired.

The next blank is boundary "D" which is the foreground disc-resident address. Move the address up to start the area on a page boundary, if desired.

The next blank is boundary "X," concerns base page linkages and requires some explanation. The boundary being moved is shown as "X" on Figure 6-5. After the real-time disc-resident programs are loaded, RTGEN reports the next available base page link above the links used. The linkage area for real-time disc-resident programs is initially established by the program loaded that requires the most links. If programs requiring more links are to be loaded on-line by the RTE-II relocating loader, this area will have to be expanded by moving the boundary up (it cannot be moved down). However, enough links must be reserved in the background disc-resident area for the background programs yet to be loaded by RTGEN.

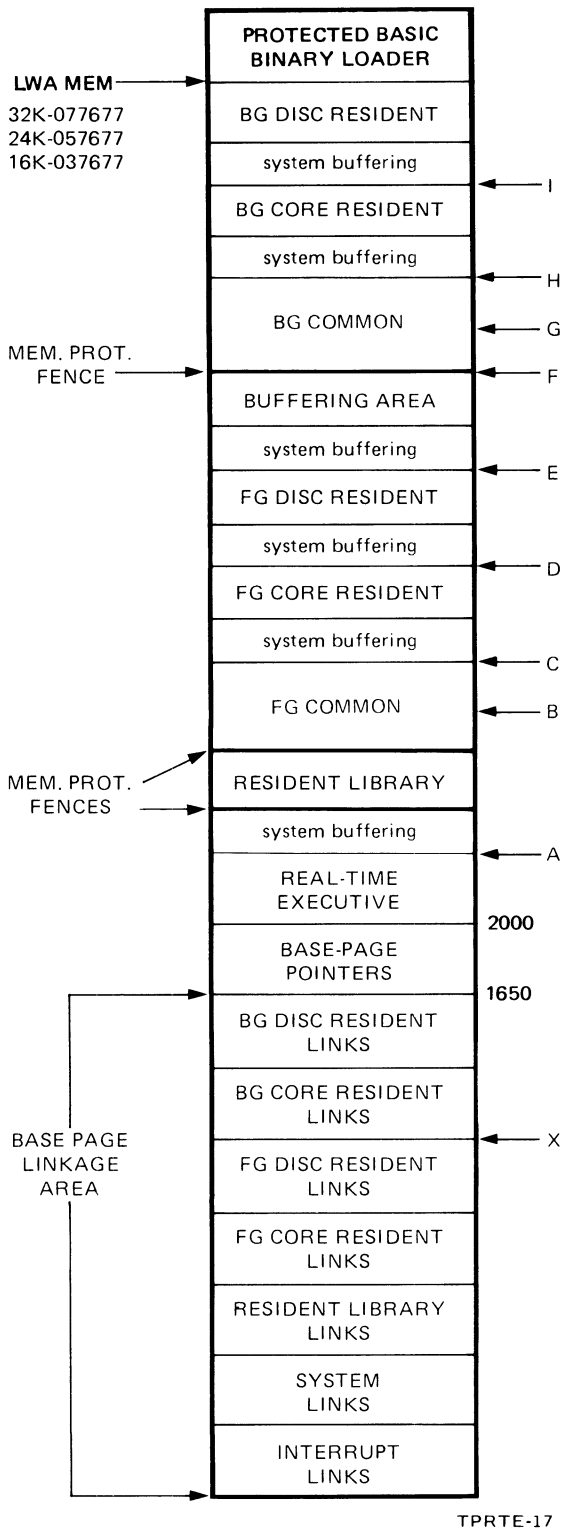


Figure 6-5. CPU Memory Allocations in a Configured RTE-II System

A recommended boundary address of octal 1100 will usually optimize the system if it is to include the usual background programs ASMB, FTN, FTN4, LOADR, EDIT, etc. These programs usually require approximately 550 octal links. The ideal boundary is one which allows RTGEN to allocate as near to 1650 links as possible. For example, if boundary X is established at 1100, and, after loading the background disc resident programs RTGEN reported the next base page linkage available as 1651, then the linkage area is as optimized as possible.

CAUTION

If RTGEN reports that more than 1650 links were used, the generation is VOID, and must be restarted at the beginning.

The next blank, shown as boundary "E," defines the real-time disc-resident area for on-line loading with the RTE-II relocating loader. This area is initially established by the largest program loaded into that area. If a larger program will be loaded into the system on-line, this area must be made larger. This boundary also affects the area reserved for re-entrant processing, buffered transfers, and for background; that is, the more area given to foreground, the less there is available for these other areas.

Boundary "F" establishes the background area used for disc-resident, and core-resident programs, plus the common area that is used by both. A recommended procedure for determining the boundaries of "E" and "F" is as follows:

a. Calculate the area needed for the largest background disc-resident program that will be used.

b. Add to this the area needed for the background core-resident programs, and the background common area.

c. Subtract this area from the last word of available memory (LWA MEM).

Example:

37677 – (11677) = 26000 (octal)

37677 = LWAM in 16K
A + B + C
A = size of largest disc-resident
B = total core resident size
C = common size

This is boundary "F" (BG BOUNDARY).

d. Subtract the area required for buffering. The amount recommended for buffering is 2000 words.

Example:

26000 – 2000 = 24000 (octal)

This is boundary "E" (response to CHANGE SYS AVMEM).

The next blank is area "G" and is the background common area. The number of words allocated for background common is reported and then RTGEN gives the user the opportunity of increasing the area.

The next blank is boundary "H" which is the background core-resident address. Move the address up to start the area on a page boundary.

The next blank is boundary "I" which is the background disc-resident address. Move the address up to start the area on a page boundary.

The final blank is for the system size in tracks and sectors. Obtain this number from the printout and write it in (if desired).

PREPARE TAPE SYSTEM

Using the Prepare Tape System as described below, the user can create a magnetic tape or fixed head disc file of the relocatable program modules used during RTGEN.

The Prepare Tape System (PTS) is a program for creating files of relocatable software modules on fixed head disc, drum, or magnetic tape units. RTGEN can then use these files to generate a configured system. When using PTS to generate a file for use during RTGEN execution, follow the instructions given in the Prepare Tape System Manual (HP Part No. 02116-91751). The instructions given in the manual for DSGEN apply directly to RTGEN.

In Section III of the PTS Manual under Operating Instructions, substitute the following information under step 1.

1. Gather all the relocatable system and user program tapes. The suggested order for loading onto the mass storage device is as follows:

Core Resident System
I/O Drivers
Power Fail (DVP 43)
System Programs written by the user
Multi-Terminal Monitor
Foreground Core-Resident Programs
Foreground Disc-Resident Programs
Background Core-Resident Programs
Assembler (Main and its Segments)
FORTRAN (Main and its Segments) and or
FORTRAN IV (Main and its Segments) but not both FORTRAN IV Versions
ALGOL
Auto Restart
Relocating Loader
Editors
Batch Monitor
Other Background Disc-Resident Programs and their respective segments, if any.
System Library
Batch Monitor Library
Library Programs
Utility Programs

NOTE

Some of the above relocatable modules may not be present in some configurations.

Table 6-9. System Configuration Worksheet

PROGRAM INPUT PHASE

CORE RESIDENT SYSTEM
I/O DRIVERS
USER'S SYSTEM PROGRAMS
FOREGROUND CORE RESIDENT PROGRAMS
FOREGROUND DISC RESIDENT PROGRAMS
BACKGROUND CORE RESIDENT PROGRAMS
BACKGROUND DISC RESIDENT PROGRAMS
AND THEIR RESPECTIVE SEGMENTS
LIBRARY PROGRAMS
UTILITY PROGRAMS
SUBROUTINES

NO UNDEF EXTS

PARAMETER INPUT PHASE

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

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

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

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

/E

CHANGE ENTS?

_____,_____,_____

_____,_____,_____

_____,_____,_____

/E

#OF BLANK ID SEGMENTS:

_____

#OF BLANK BG SEG. ID SEGMENTS?

_____

FWA BP LINKAGE?

_____

TABLE GENERATION PHASE

*#OF I/O CLASSES?

_____

* # OF LU MAPPINGS?

_____

* # OF RESOURCE NUMBERS?

_____

* BUFFER LIMITS (LOW, HIGH)?

_____,_____

* EQUIPMENT TABLE ENTRY

EQT 01?

_____,DV____,_____,_____,T=____,X=____

EQT 02?

_____,DV____,_____,_____,T=____,X=____

EQT 03?

_____,DV____,_____,_____,T=____,X=____

EQT 04?

_____,DV____,_____,_____,T=____,X=____

EQT 05?

_____,DV____,_____,_____,T=____,X=____

EQT 06?

_____,DV____,_____,_____,T=____,X=____

EQT 07?

_____,DV____,_____,_____,T=____,X=____

EQT 08?

_____,DV____,_____,_____,T=____,X=____

EQT 09?

_____,DV____,_____,_____,T=____,X=____

EQT 10?

_____,DV____,_____,_____,T=____,X=____

/E

TPRTE-22a

Table 6-9. System Configuration Worksheet (continued)

*** DEVICE REFERENCE TABLE**

1 = EQT #? (SYSTEM TELEPRINTER)

_____, _____

2 = EQT #? (SYSTEM MASS STORAGE)

_____, _____

3 = EQT #? (AUXILIARY MASS STORAGE)

_____, _____

4 = EQT #? (STANDARD PUNCH UNIT)

_____, _____

5 = EQT #? (STANDARD INPUT UNIT)

_____, _____

6 = EQT #? (STANDARD LIST UNIT)

_____, _____

7 = EQT #?

_____, _____

8 = EQT #? (MAG TAPE RECOMMENDED)

_____, _____

9 = EQT #?

_____, _____

10 = EQT #?

_____, _____

11 = EQT #?

_____, _____

12 = EQT #?

_____, _____

13 = EQT #I

_____, _____

14 = EQT #?

_____, _____

15 = EQT #?

_____, _____

/E

*** INTERRUPT TABLE**

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

_____, _____, _____

/E

TPRTE-22b

Table 6-9. System Configuration Worksheet (continued)

SYSTEM BOUNDARIES PHASE

LIB ADDRS _____ (A)

CHANGE LIB ADDRS?

_____

FG COMMON _____ (B)

CHANGE FG COMMON?

_____

FG RES ADD _____ (C)

CHANGE FG RES ADD?

_____

FG DSC ADD _____ (D)

CHANGE FG DSC ADD?

_____

BP LINKAGE _____ (X)

CHANGE BP LINKAGE?

_____

SYS AVMEM _____ (E)

CHANGE SYS AVMEM?

_____

BG BOUNDRY _____ (F)

CHANGE BG BOUNDRY?

_____

BG COMMON _____ (G)

CHANGE BG COMMON?

_____

BG RES ADD _____ (H)

CHANGE BG RES ADD?

_____

BG DSC ADD _____ (I)

CHANGE BG DSC ADD?

_____

SYSTEM STORED ON DISC

SYS SIZE: _____ TRKS, _____ SECS (10)

TPRTE-22c

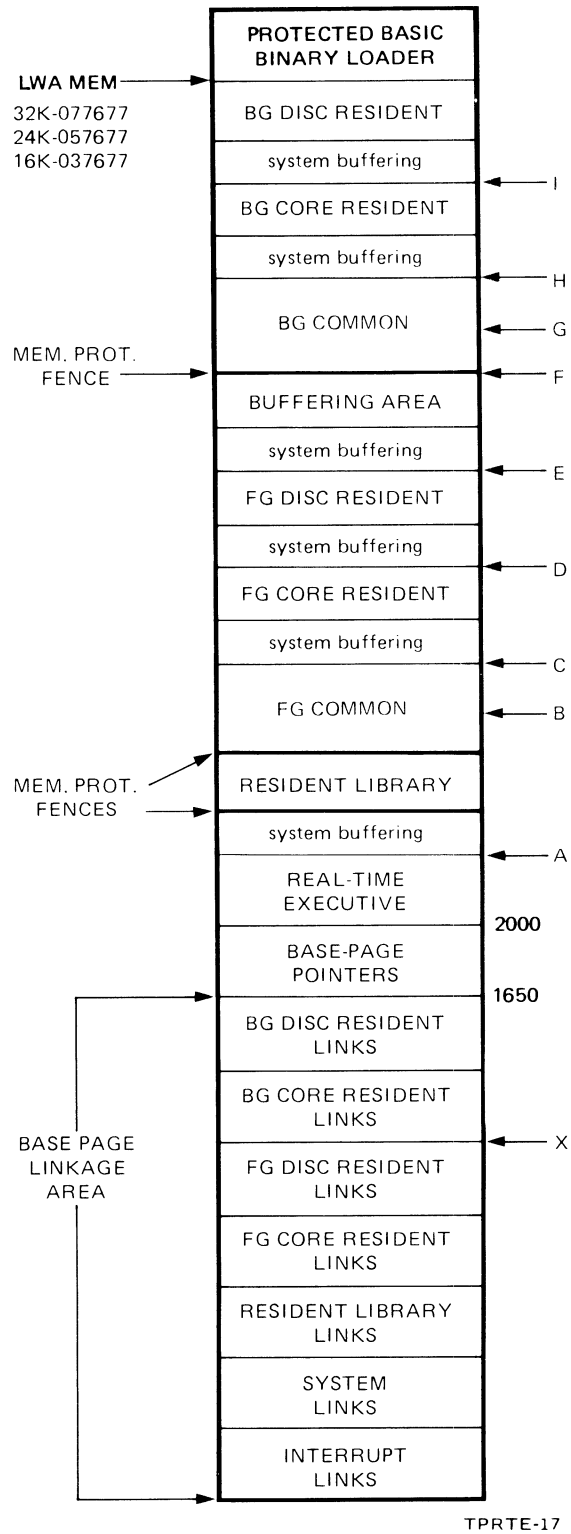


Figure 6-5. (repeated) CPU Memory Allocations in a Configured RTE-II System

PART 2
Moving Head System Generation

INTRODUCTION

The set up and operation of a Moving Head RTE-II System involves two essential steps; the system must be configured using the Real-Time System Generator, RTGEN and it must be initiated from the disc by a Basic Binary Disc Loader (BBDL) or by a bootstrap tape that is punched during RTGEN. For an HP 21MX Computer, use the appropriate ROM loader.

This part describes the steps necessary to configure an RTE-II System, with the absolute binary code based on a moving head disc drive. All of the required information should be pre-planned and located on the Configuration Worksheets filled out in Part 1. The answers used in this part are from an example RTE-II System that has been configured on worksheets per the instructions in Part 1. The RTGEN listing is located in Appendix C.

MOVING HEAD DISC RTGEN

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

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

During generation based on the HP 7900 disc drive, RTGEN will report defective tracks and initialize them defective. Up to ten bad tracks are allowed in the system before RTGEN aborts. The system cannot operate if any bad tracks are located within the area established for the absolute code on LU2.

During generation based on the HP 7905 disc drive, RTGEN will report defective tracks, initialize them as defective, and assign spare tracks as long as spare tracks are available for that subchannel. If there are not enough spare tracks available, RTGEN issues the error message ERR43 and then restarts the initialization phase.

The system is limited to the disc tracks specified during RTGEN. As a result, the user can create many different configurations of RTE-II Systems, all coresiding on the same moving head disc subchannels.

RTGEN is an absolute program, loaded into main memory by the Basic Binary Disc Loader (BBDL) from paper tape. For an HP 21MX Computer, use the appropriate ROM loader. Since RTGEN is independent of the system which it generates, the I/O operations of RTGEN require SIO Drivers.

MULTIPLE CPU/7905 SYSTEMS

The HP 7905 versions of RTGEN, the bootstrap loader, and the on-line driver support multiple CPU operation. More than one CPU can share one or more disc drives under the following conditions:

- The system area (that is, LU2 and LU3) for one CPU cannot occupy the same system disc tracks as that of another CPU.
- Systems may map tracks in the same peripheral disc area. However, they should share access to these areas only as described in Appendix B under Multiple CPU/7905 System Operation.
- The generator should not be allowed to use as scratch or to initialize areas of the disc already in use by any other CPU.

As an aid to using a multiple CPU system, it is recommended that the disc track map be identical for each CPU. Further, logical unit numbers should not be assigned to subchannels already assigned to another CPU.

OPERATING PROCEDURES

The operation of RTGEN involves five phases:

a. INITIALIZATION PHASE. RTGEN requests specifications for the system, including a track map of system disc space, memory, time base generator channel, swapping option, and program input devices.

b. PROGRAM INPUT PHASE. The operator loads the relocatable programs provided with the system, and created by the user.

c. PARAMETER INPUT PHASE. Parameters describing or changing the type, priority, and execution interval of each program may be entered. (Although this information may already be included in the program's NAM record, it can be changed at this point.) Entry points are changed and a number of blank ID segments are reserved for subsequent program addition. RTGEN requests a specification of the base page linkage and begins loading programs onto the disc.

d. TABLE GENERATION PHASE. RTGEN requests information about Class I/O, LU switching, Resource Numbers and buffer limits. RTGEN then requests information for the equipment table, device reference table, and interrupt table and finishes loading the rest of the programs onto the disc.

e. SYSTEM BOUNDARIES PHASE. RTGEN begins loading the system and stops at each boundary and reports the address. If the user desires, the address can be moved up to a page boundary. The gap between the two then reverts to system use.

NOTE

During the first three phases, RTGEN can be restarted at octal address 100. The restarting points are back to the beginning of the Initialization Phase, Program Input Phase or Parameter Input Phase. Make certain that the switch register options are set up before pressing RUN.

To execute RTGEN and configure an RTE-II System, follow these steps:

a. Turn on the system using the main power switch. Ensure that the peripheral equipment power switches are on and that the operating mode is on-line.

NOTE

For the 7905 disc drive, set the hardware unit switch to the unit number written on your worksheet.

b. Insert a disc cartridge into the disc drive.
c. For instructions on how to disable the disc protect mechanism for the discs to be configured during this generation process, refer to the appropriate disc drive reference manual:

HP 7900A Disc Drive Operating and Service Manual, Part No. 07900-90002.

HP 7905A Disc Drive Operator's Manual, Part No. 07905-90009

d. For instructions on how to load programs from paper tape via the system photoreader, refer to the appropriate HP computer hardware reference manual:

HP 2100A Computer Reference Manual, Part No. 02100-90001, under "Operating Procedures for Operator Panel."

HP 2100S Microprogrammable Systems Computer Reference Manual, Part No. 02100-90160, under "Operating Procedures for Operator Panel."

HP 21MX Computer Reference Manual, Part No. 02108-90002, under "Basic Operating Examples."

e. Load the RTGEN program from paper tape.
f. Load configured SIO drivers from paper tape (or load and configure SIO drivers individually).
g. Refer to Table 6-10 in this section and set the switch register to the desired initial options. Most options can be set at the beginning of the generation process even though they are not used immediately.

NOTE

During the Program Input Phase, switch register bit 15 may need to be changed if an error occurs, and bits 0 and 1 require resetting at least once.

h. If the answers to RTGEN requests are contained on paper tape, place that tape in the photoreader. Prepare the photoreader for loading.
i. Set the RTGEN program starting address to 100 (octal).
j. Press RUN.

RTGEN begins the generation process at the Initialization Phase.

Table 6-10. Switch Register Options

Bit	Function	When to Set
15	1 = The program just read will be purged. Input data is ignored until the next NAM record. The same program starting from the beginning or the next program will then be read. 0 = The program just read is not purged. The last record can be re-read.	After error 2, 3, or 4 during the Program Input Phase
	Print the entry point list.	Before answering FWA BP LINKAGE? (Change anytime)
14	Establishes current page-linking mode.	Before answering FWA BP LINKAGE? (Change anytime)
13	Print base page linkage listing.	Before answering FWA BP LINKAGE? (Change anytime)
6	Print only errors on list device. The generation printout is omitted.	Anytime
5	Answers are read from the tape reader instead of the teletype. If an input error occurs, control is automatically returned to the teletype for the next command only. If the error is meaningless (e.g., an unwanted routine is referenced), input from the tape reader can be resumed by entering an asterisk (followed by any comments if desired) and carriage return, line feed. The switch can be turned off at this time in order to enter multiple inputs from the TTY.	Beginning or during generation
4	Questions and answers are punched on the punch device.	Beginning or during generation
3	Questions and answers are printed on the list device. This assumes the list device is a line printer because RTGEN will list to only one device.	Beginning or during generation
1 0	Switches 1 and 0 are set as follows: 1 0 Meaning 0 0 Load from program input unit 1 0 Load from library input unit 0 1 Print list of undefined externals and halt. 1 1 If still set terminate program Input Phase.	Beginning of generation or during Program Input Phase.

NOTES

1. When answers are from the photoreader, and an error occurs, the generator transfers control to the keyboard automatically. Type in the correct answer or a comment to continue (ignoring the error).
2. Comments must be preceded with an asterisk. A comment may be included within an input line following a legal response to the system, or entered as a stand-alone comment line. For example, the following are both legal comment lines:

   * END OF PHASE

   /E * END OF PHASE
3. The switch setting is examined each time the action it controls is started. Thus the setting may be examined at any time.

INITIALIZATION PHASE

During the initialization phase, RTGEN first requests information necessary to generate a disc track map that defines which tracks of each platter are assigned to the system. Once the track map is established, RTGEN goes on to request more information necessary to generate the system. 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 CR, LF is entered, type the RUBOUT key then CR/LF. Otherwise, restart at step "e" above.

The following sample dialog is from a system being generated on an HP 7900/7901 disc. For a system being generated on an HP 7905 disc, the first two RTGEN prompts differ. Refer back to Table 6-6 and 6-7 for the difference.

RTGEN requests the higher priority select code (octal) of the system disc controller

MH DISC CHNL?
21

RTGEN requests the starting track and number of tracks (decimal) of each subchannel that will be assigned to the system. Up to eight track assignments can be entered, one for each existing subchannel. The even numbered subchannels are the fixed platters and the odd numbered subchannels are the removable platters (i.e., subchannel 0 is the fixed platter and subchannel 1 is the removable platter of the first drive.

#TRKS, FIRST TRK ON SUBCHNL:
0?

Operator responds with the decimal number of tracks and starting track number for subchannel 0. If there are to be no tracks from subchannel 0 assigned to the system, enter a 0. RTGEN continues to request the track assignments for each subchannel up to seven or until /E is entered. Refer to the Configuration Worksheet and enter the correct numbers. The Appendix C example is entered as follows:

0?
203,0
1?
203,0
2?
203,0
3?
203,0
4?
100,50
5?
203,0
6?
203,0
7?
203,0

RTGEN requests the number of 128 word sectors (decimal) per logical track on the system disc.

128 WORD SECTORS/TRACK?
48

NOTE

The following two questions concerning system subchannel and scratch disc subchannel are not asked if only one subchannel is assigned to the system.

RTGEN requests the subchannel number of the system disc (LU2). This is the disc that the absolute code will be stored upon and can be any one of the subchannels assigned to the system. The operator responds with a subchannel number (from the worksheet) that contains enough tracks for the absolute code.

SYSTEM SUBCHNL?
0

RTGEN requests the subchannel number of the scratch disc. This is the area required for the relocatable modules used to build the system, and can be any one of the subchannels assigned to the system. It is recommended that the scratch area not be located on the system subchannel. Then the absolute area cannot overlay the relocatable modules before they are used.

SCRATCH SUBCHNL?
0

RTGEN asks if there is to be an auxiliary disc (LU3) and its number of tracks. Answer YES if there is an auxiliary disc and it's on the same controller as the system disc. Answer NO if there is no auxiliary disc. Answer with the number of tracks if there is an auxiliary disc and its not on the same controller as the system disc.

AUX DISC (YES OR NO OR # OF TRKS)?
YES

If the above question was answered with a number denoting how many tracks on the auxiliary disc (on a different controller than the system disc), then the next question asked is:

128 WORD SECTORS/TRACK?

Answer accordingly (decimal).

If the answer to the AUX DISC? question was NO, then RTGEN skips to the START SCRATCH? question. If the answer to the AUX DISC? question was YES, then RTGEN requests the auxiliary disc subchannel number. The operator responds with a valid subchannel number (not the system subchannel).

AUX DISC SUBCHNL?
1

RTGEN requests the track number starting the disc scratch area. The operator responds with a decimal relative track number. For example, subchannel 4 has tracks addressed 50 to 150 available. To start the scratch area on the first available track (50) enter a zero (0). To start the scratch at track 75, enter 25. Note that if the scratch subchannel is the same as the system subchannel, entering a zero defaults the start scratch to the midpoint of the available disc space on the system subchannel.

START SCRATCH?
0

RTGEN requests the select code of the time base generator (octal).

TBG CHNL?
13

RTGEN requests the address of the privileged interrupt I/O card (if present).

PRIV. INT. CARD ADDR?

Operator responds with the octal select code of the privileged interrupt HP 12620 card (and all devices in higher priority slots become privileged) or zero if the card is not used.

12

RTGEN asks if swapping is allowed in the foreground area. Answer YES or NO.

FG SWAPPING?
YES

RTGEN asks if swapping is allowed in the background area. Answer YES or NO.

BG SWAPPING?
YES

RTGEN asks if any program is allowed to be locked into the foreground. Answer YES or NO.

FG CORE LOCK?
YES

RTGEN asks if any program is allowed to be locked into the background. Answer YES or NO.

BG CORE LOCK?
YES

RTGEN next requests the amount of swap delay time. Enter a decimal number between 0 and 255 (meaning tens of milliseconds).

SWAP DELAY?
50

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

LWA MEM?

Operator responds with 37677 for 16K, 57677 for 24K, or 77677 for 32K.

77677

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

PRGM INPT?

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

MT

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

LIBR INPT?

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

PT

NOTE

Any type of program can be entered through the program input unit or the library input unit.

RTGEN requests the type of input unit for parameters describing the relocatable programs.

PRAM INPT?

Operator responds with PT OR TY.

TY

RTGEN asks if the operator wishes to initialize disc subchannels other than the system, auxiliary, and scratch subchannel. RTGEN asks this question only for those subchannels assigned to the system but not declared as LU2, LU3, or scratch, these subchannels being automatically initialized by RTGEN. Operator responds with a YES or NO. If the disc is new or has any write protect flags written on it, it must be initialized. If the disc has data stored on it in the system designated area, and the user does not want to disturb it, the answer is NO.

INITIALIZE SUBCHNL:
2?
NO
3?
NO
4?
NO
5?
NO
6?
NO
7?
NO

Next RTGEN checks the hardware disc protect switch. If the disc is protected RTGEN prints

TURN OFF DISC PROTECT – PRESS RUN

and halts with the Memory Data Register = 102032. The operator must turn off the disc protect switch and press RUN to continue to the next phase. If the switch is already off the above message is not printed and RTGEN proceeds to the PUNCH BOOT question. If the response to PGM INPT or LIB INPT was MT, the magnetic tape unit will rewind to the load point, and then space forward to relocatable file number two. A HALT 102044 indicates that the magnetic tape unit is not ready. Set it up and press RUN.

RTGEN asks if a paper tape bootstrap is to be punched. If the RTE-II System is located on subchannel 0 or 1 and starts on track 0, the bootstrap tape is not required. Otherwise the first answer should be YES. Note that the bootstrap is unique to the absolute system and first track number only. RTGEN keeps repeating the question until NO is entered. In this fashion the user can punch as many bootstraps as he feels he needs.

PUNCH BOOT?
YES
PUNCH BOOT?
NO

PROGRAM INPUT PHASE

During the program input phase, RTGEN accepts relocatable programs from the program input unit and library input unit specified during the initialization phase. The operator selects the input device by setting switch register bits 0-1.

00	=	program input unit
10	=	library input unit
01	=	print list of undefined externals, or after the printout
01	=	terminate input phase

If an error is detected during the Program Input Phase, the name of the offending program is printed. At this point the operator can set or clear bit 15 before pushing RUN to accomplish one of the following:

Bit 15 = 1 The program just read will be purged. Input data is ignored until the next NAM record. The next program will then be read.

Bit 15 = 0 The program just read is not purged. The last record can be re-read.

Relocatable programs should be loaded in the following order. Note that some of the programs may not be present in some configurations.

Core Resident System
I/O Drivers
Power Fail (DVP 43)
System Programs written by the user
Multi-Terminal Monitor

Foreground Core-Resident Programs
Foreground Disc-Resident Programs
Background Core-Resident Programs
Assembler (Main and its Segments)
FORTRAN (Main and its Segments) and/or
FORTRAN IV (Main and its Segments) but not both FORTRAN IV Versions
ALGOL
Auto Restart
Relocating Loader
Editors
Batch Monitor
Other Background Disc-Resident Programs and their respective segments, if any.
System Library
Batch Monitor Library
Library Programs
Utility Programs

NOTE

If a program is being loaded from paper tape, and was not generated with the type code in the NAM record, the operator can modify it during the Parameter Input Phase.

The operator presses RUN. After a program is loaded, the message "*EOT" is printed whenever an end-of-file or end-of-tape occurs. The computer halts.

At this point, the operator has several alternatives:

a. Additional programs can be loaded through the same device by pushing RUN.
b. Input can be switched to the other input device by setting the switch register bits to binary 00 or 10.
c. After each *EOT message, a list of all undefined externals can be printed by setting the switch register bits to binary 01 and pushing RUN. At this point the operator can reset the switches to point to the desired input device, and load additional routines needed to satisfy any undefined externals. If there are none, the message NO UNDEF EXTS is printed and the computer executes a HALT 77. To continue loading programs, reset the switch register to binary 00, or 10, place the program in the input device, and push RUN.
d. To terminate the program input phase, set the switch register to binary 01 and push RUN. (RUN must be pushed again after NO UNDEF EXTS is printed.) If magnetic tape is used for the program input, it will rewind and go off-line at this point (after the second RUN).
e. To restart this phase go to the starting address of the program, octal 100, and push RUN.

PARAMETER INPUT PHASE

If the teletype was not specified as the PRAM INPT device during initialization, the computer executes a HALT 77 to wait for the parameter tape to be inserted in the photoreader. Push RUN to continue. If there are any errors on the parameter tape, they will be printed on the list device.

During the parameter input phase, the operator can modify the type, priority, or execution intervals (in decimal) of any of the programs entered during the program input phase (except that the program type code of background main programs and their segments cannot be changed without losing their relationship to each other).

Each parameter record is of this general form:

name, type [*,priority*] [*,execution interval*]

Refer to the Configuration Worksheet and enter any modification parameters that are listed. If there are none, enter a /E on the teletype. As an example (see Appendix C) the operator can respond with:

```
PARAMETERS
#BSC,14
WHZAT,2,10
MEM,1,32767
SYSON,82,32766
RNRQ,14
LURQ,14
EQLU,14
/E
```

NOTE

Refer to Part 1, Parameter Input Phase for an explanation of type codes 81 through 92.

RTGEN requests if there are any entry records to be changed. The numbers are assumed octal unless followed by a "D" which signifies decimal.

```
CHANGE ENTS?

*EQU MACRO'S
.MPY,RP,100200
.DIV,RP,100400
.DLD,RP,104200
.DST,RP,104400
```

```
*HFP MACRO'S
.FAD,RP,105000
.FSB,RP,105020
.FMP,RP,105040
.FDV,RP,105060
IFIX,RP,105100
FLOAT,RP,105120
*FFP MACRO'S
DBLE,RP,105201
SNGL,RP,105202
.XMPY,RP,105203
.XDIV,RP,105204
.DFER,RP,105205
.XADD,RP,105213
.XSUB,RP,105214
.GOTO,RP,105221
..MAP,RP,105222
.ENTR,RP,105223
.ENTP,RP,105224
/E
```

RTGEN requests the number of blank 28-word ID segments to be allocated for on-line loading of programs by the relocating loader. The operator responds with a one or two digit decimal number (zero is changed to one, because one is required to do any on-line loading); 29 words are reserved in the resident table area for each blank ID segment.

```
# OF BLANK ID SEGMENTS?
10
```

RTGEN next requests the number of blank 9-word ID segments to be allocated for background segments. The operator responds with a one or two digit decimal number (zero is legal); 10 words are reserved in the resident table area for each blank BG ID segment.

```
# OF BLANK BG SEG. ID SEGMENTS?
25
```

Before the next question concerning Base Page Linkages is answered, switch 14 should be set or cleared according to the following directions. With bit 14 set, RTGEN, as each module is loaded, creates at each end of a module which crosses a page boundary, a current page-link buffer. The current page-link buffer is used for DBL records that reference addresses across the current page boundary. If this current page-link buffer overflows, then RTGEN reverts to using base page links for the module. The end result is that base page is saved as links are established in the current page buffer.

NOTE

EXT references and Assembly Language type 3 or 5 programs still use exclusively base page for establishing links. However, this is not true for subroutines referenced by these programs.

RTGEN requests the first word of available core memory in base page. Operator responds with the first available octal select code number after the last I/O card.

```
FWA BP LINKAGE?
100
```

Disc loading begins with the modules of the system, including I/O drivers. As RTGEN loads these programs, it prints SYSTEM, followed by a memory map giving the starting locations and, if switch register bit 15 is set, the entry points for all main programs and subroutines (subroutines are indented two spaces). Also, if bit 13 is set, the base page linkage is reported after each module is loaded.

TABLE GENERATION PHASE

After the last system module is loaded RTGEN requests how many Class I/O numbers are to be allocated.

```
*# OF I/O CLASSES?
16
```

RTGEN next requests the maximum number of logical unit numbers referred to in a single job within the Batch and Spool Monitor.

```
*# of LU MAPPINGS?
8
```

RTGEN next requests how many Resource Numbers are to be allocated.

```
*# OF RESOURCE NUMBERS?
32
```

RTGEN next requests the buffer limits.

```
BUFFER LIMITS (LOW,HIGH)?
100,400
```

Next, RTGEN generates the three I/O tables: Equipment table, device reference table, and the interrupt table.

RTGEN requests the equipment table entries

*EQUIPMENT TABLE ENTRY

Operator responds with a series of EQT entries, which are assigned EQT numbers sequentially from one as they are entered. The EQT entry relates the EQT number to an I/O channel and driver. Refer to the Configuration Worksheet and enter the Equipment table entries. The Appendix C example is entered as follows:

EQT 01?
21,DVR31,D

EQT 02?
15,DVR00,B,T=32767

EQT 03?
10,DVR50

EQT 04?
17,DVR02,B,T=32767

EQT 05?
16,DVR01,T=32767

EQT 06?
20,DVR12,B,T=32767

EQT 07?
25,DVR00,B,T=32767

EQT 08?
23,DVR23,D,B,T = 32767

EQT 09?
26,DVR62,D

EQT 10?
27,DVR61

EQT 11?
14,DVR11,D

EQT 12?
30,DVR00,B,T=32767

EQT 13?
31,DVR00,B,T=32767

EQT 14?
32,DVR00,B,T=32767

EQT 15?
33,DVR00,B,T=32767

EQT 16?
34,DVR00,B,T=32767

EQT 17?
35,DVR00,B,T=32767

EQT 18?
36,DVR00,B,T=32767

EQT 19?
37,DVR00,B,T=32767

EQT 20?
72,DVS43,X=18

EQT 21?
73,DVS43,X=18

EQT 22?
74,DVS43,X=18

EQT 23?
75,DVS43,X=18

EQT 24?
76,DVS43,X=18

EQT 25?
77,DVS43,X=18

EQT 26?
4,DVP43

EQT 27?
/E

RTGEN requests the logical unit assignments for the device reference table.

*DEVICE REFERENCE TABLE

For each logical unit number, RTGEN prints

n=EQT#?

where *n* is a decimal integer starting with one.

Operator responds with an EQT entry number appropriate to the standard definition of *n* and the subchannel number if appropriate. Logical unit numbers 1 through 6 are predefined in the system as:

0 – bit bucket (no action required)
1 – system teleprinter
2 – system mass storage
3 – auxiliary mass storage
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:

```
* DEVICE REFERENCE TABLE

 1 = EQT #?
 2,0,                    SYSTEM CONSOLE

 2 = EQT #?
 1,0,                    SYSTEM DISC

 3 = EQT #?
 1,1,                    AUXILIARY DISC

 4 = EQT #?
 4,4,                    PAPER TAPE PUNCH

 5 = EQT #?
 5,0,                    PHOTOREADER

 6 = EQT #?
 6,0,                    LINE PRINTER

 7 = EQT #?
 7,0,                    BACKGROUND TERMINAL

 8 = EQT #?
 8,0,                    MAGNETIC TAPE, UNIT 0

 9 = EQT #?
 8,1,                    MAG TAPE, UNIT 1

10 = EQT #?
 8,2,                    MAG TAPE, UNIT 2

11 = EQT #?
 8,3,                    MAG TAPE, UNIT 3

12 = EQT #?
 11,0,                   CARD READER

13 = EQT #?
 0                       BIT BUCKET

14 = EQT #?
 1,2,                    PERIPHERAL DISC

15 = EQT #?
 1,3,                    PERIPHERAL DISC

16 = EQT #?
 1,4,                    PERIPHERAL DISC

17 = EQT #?
 1,5,                    PERIPHERAL DISC

18 = EQT #?
 1,6,                    PERIPHERAL DISC

19 = EQT #?
 1,7,                    PERIPHERAL DISC

20 = EQT #?
 12,0,                   TERMINAL

21 = EQT #?
 13,0,                   TERMINAL

22 = EQT #?
 14,0,                   TERMINAL

23 = EQT #?
 15,0,                   TERMINAL

24 = EQT #?
 16,0,                   TERMINAL

25 = EQT #?
 17,0,                   TERMINAL

26 = EQT #?
 18,0,                   TERMINAL

27 = EQT #?
 19,0,                   TERMINAL

28 = EQT #?
 0

29 = EQT #?
 0

30 = EQT #?
 9,0                     H.P. 2313B SUBSYSTEM

31 = EQT #?
 10,0,                   H.P. 6940 SUBSYSTEM

32 = EQT #?
 0

33 =EQT #?
 0

34 = EQT #?
 0

35 = EQT #?
 0

36 = EQT #?
 0

37 = EQT #?
 0

38 = EQT #?
 0
```

```
39 = EQT #?
 0

40 = EQT #?
 3,0,                    RDTS SUBSYSTEM

41 = EQT #?
 0

42 = EQT #?
 0

43 = EQT #?
 0

44 = EQT #?
 0

45 = EQT #?
 0

46 = EQT #?
 0

47 = EQT #?
 0

48 = EQT #?
 0

49 = EQT #?
 0

50 = EQT #?
 0

51 = EQT #?
 20,0,                   SPOOL LU

52 = EQT #?
 21,0,                   SPOOL LU

53 = EQT #?
 22,0,                   SPOOL LU

54 = EQT #
 23,0,                   SPOOL LU

55 = EQT #?
 24,0,                   SPOOL LU

56 = EQT #?
 25,0,                   SPOOL LU

57 = EQT #?
 26,0,                   POWER FAIL

58 = EQT #?
/E
```

RTGEN requests the interrupt table entries.

*INTERRUPT TABLE

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 is entered as follows:

```
* INTERRUPT TABLE
 4,ENT,SPOWR
 10,EQT,3
 11,EQT,3
 14,EQT,11
 15,PRG,PRMPT
 16,EQT,5
 17,EQT,4
 20,EQT,6
 21,EQT,1
 22,EQT,1
 23,EQT,8
 24,EQT,8
 25,PRG,PRMPT
 26,EQT,9
 27,EQT,10
 30,PRG,PRMPT
 31,PRG,PRMPT
 32,PRG,PRMPT
 33,PRG,PRMPT
 34,PRG,PRMPT
 35,PRG,PRMPT
 36,PRG,PRMPT
 37,PRG,PRMPT
 72,EQT,20
 73,EQT,21
 74,EQT,22
 75,EQT,23
 76,EQT,24
 77,EQT,25
 /E
```

SYSTEM BOUNDARIES PHASE

When the interrupt table is completed RTGEN prints the new first word of available core memory in base page.

BP LINKAGE 00250

RTGEN reports the starting address of the library and waits for any change. Enter 0 for no change.

```
LIB ADDRS 34761
CHANGE LIB ADDRS?
 36000
```

RTGEN prints LIBRARY, followed by names (and if bit 15 is set, entry point addresses) of all routines which are referenced by real-time and background programs. After this RTGEN reports the new first word of available core memory in base page.

BP LINKAGE 00431

RTGEN reports the total number of words allocated to foreground common and waits for any change. Enter 0 for no change.

FG COMMON 00000
CHANGE FG COMMON?
200

RTGEN reports the starting address of the foreground core resident area and waits for any change. Enter 0 for no change.

FG RES ADD 44013
CHANGE FG RES ADD?
0

At this time, if any type 9 (foreground core-resident program that uses background common) or type 10 (foreground disc-resident program that uses background common) programs are in the system, RTGEN requests the background boundary. This is the start of background common. Refer to Part 1 for guidelines in determining this entry.

BG BOUNDRY 53600
CHANGE BG BOUNDRY?
0

Next RTGEN prints FG RESIDENTS, and loads the real-time core resident programs and reports the new first word of available core memory in base page.

BP LINKAGE 00460

NOTE

If there are no library or real-time core resident programs, RTGEN prints (NONE) and does not report the links because they have not changed.

RTGEN reports the starting address of foreground disc resident area and waits for any change. Enter 0 for no change.

FG DSC ADD 45600
CHANGE FG DSC ADD?
46000

RTGEN prints FG DISC RESIDENTS, and loads the real-time disc-resident programs, reports the new first word of available core memory in base page, and waits for any change. Increasing the base page linkage area here allows for future additions of larger real-time disc-resident programs on-line with the RTE-II Relocating Loader.

BP LINKAGE 00631
CHANGE BP LINKAGE?

Operator responds with zero for no change, or an octal address greater than the last base page link reported. Refer to Part 1 for guidelines in determining this entry. The Appendix C entry is:

740

RTGEN next reports the first word address of the system available memory and waits for a change. Enter a 0 for no change or an octal address greater than the number reported. Increasing this number allows more room for on-line additions with the Loader.

SYS AVMEM 53527
CHANGE SYS AVMEM?
53600

Next, if there are no type 9 or 10 programs in the system, RTGEN reports the first address of the background area. This establishes the area between the SYS AVMEM and the background and is used for temporary storage of output buffers, and re-entrant temporaries; it should be large enough to handle the largest anticipated buffered I/O transfer. Refer to Part 1 for guidelines in determining this entry. If there are type 9 or 10 programs in the system, this question was asked earlier and is not repeated here.

BG BOUNDRY 53600
CHANGE BG BOUNDRY?
0

RTGEN reports the total number of words allocated to background common and waits for any change. Enter 0 for no change.

BG COMMON 00000
CHANGE BG COMMON?
200

RTGEN reports the new background common size (unless it is zero).

BG COM 00200

RTGEN reports the starting address of the background core-resident area and waits for any change. Enter 0 for no change.

BG RES ADD 54000
CHANGE BG RES ADD?
0

Next RTGEN prints BG RESIDENTS, and loads the background resident programs. If there are no background core-resident programs RTGEN prints (NONE) and does not report the base page links because they have not changed.

RTGEN reports the starting address of the background disc-resident area and waits for any change. Enter 0 for no change.

```
BG DSC  ADD 54000
CHANGE BG DSC ADD?
 0
```

RTGEN prints BG DISC RESIDENTS, proceeds to load the background disc-resident programs, and prints names and entry points for mains and subroutines. RTGEN completes the loading and gives the total base page links used.

BP LINKAGE 01427

RTGEN then reports the system is loaded on the disc by printing

SYSTEM STORED ON DISC

RTGEN reports the system size by printing the numbers of tracks and sectors used (in decimal).

SYS SIZE: 25 TRKS, 032 SECS (10)

INITIATING RTE-II FROM THE MOVING HEAD DISC

To start up the newly configured RTE-II system, follow these steps:

a. Enable any disc protection features that were disabled for system generation. For instructions on how to enable the disc protect mechanism, refer to the appropriate disc drive reference manual:

HP 7900A Disc Drive Operating and Service Manual, Part No. 07900-90002.

HP 7905A Disc Drive Operator's Manual, Part No. 07905-90009

b. Load the configured system into main memory from disc storage. For instructions on how to load from disc, refer to the appropriate HP computer hardware reference manual:

HP 2100A Computer Reference Manual, Part No. 02100-90001, under "Operating Procedures for Operator Panel."

HP 2100S Microprogrammable Systems Computer Reference Manual, Part No. 02100-90160, under "Operating Procedures for Operator Panel."

HP 21MX Computer Reference Manual, Part No. 02108-90002, under "Basic Operating Examples."

c. When the computer halts after the disc loader has executed, press RUN.

d. The system displays the message SET TIME on the console.

e. Either set the real-time clock to the current time (see the TM operator request in Section II) or enter any other legal request (the system will set the clock).

ERROR HALTS

The following halts can occur during use of the bootstrap.

Halt Code	Cause	Recovery Action
102011	Disc error status is in the A-Register.	Check that the disc drive is ready—push RUN to retry.
102031	Same as above. Occurs during execution of disc-resident part of bootstrap.	Check that the disc drive is ready—push RUN to retry.

The following halts can occur during the operation of RTGEN.

102000	System error	Irrecoverable
102001	Magnetic tape not ready.	Check that the magnetic tape is ready—push RUN to retry.
102032	Disc protect switch in protect position.	Turn off switch and press RUN.
102044	Magnetic tape not ready.	Check that the magnetic tape is ready—push RUN to retry.
102066	Punch error (EOT)	Check type punch
102077	Normal halt. Additional data required.	Set switch register and push RUN.

MH RTGEN ERROR MESSAGES

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

MESSAGES DURING INITIALIZATION AND INPUT PHASE

ERR 01

Meaning: Invalid response to initialization request.

Action: Message is repeated. Enter valid reply.

ERR 02

Meaning: Checksum error on program input.

Action: Computer halts; reposition tape to beginning of record and press RUN to reread. If input is from MT or DF, it is automatically backspaced when RUN is pressed unless bit 15 is set.

ERR 03

Meaning: Record out of sequence.

Action: Same as ERR 02.

ERR 04

Meaning: Illegal record type.

Action: Same as ERR 02.

ERR 05

Meaning: Duplicate entry point.

Action: Revise program by re-labeling the entry points (the current entry point replaces the previous entry point).

ERR 06

Not used

ERR 07

Meaning: Program name or entry point table overflow of available memory.

Action: Irrecoverable error. Revise or delete programs.

ERR 08
name

Meaning: Duplicate program name.

Action: The current program replaces the previous program.

MESSAGES DURING THE PARAMETER PHASE

ERR 09

Meaning: Parameter name error (no such program).

Action: Enter valid parameter statement.

ERR 10

Meaning: Parameter type error.

Action: Same as ERR 09.

ERR 11

Meaning: Parameter priority error.

Action: Same as ERR 09.

ERR 12

Meaning: Execution interval error.

Action: Same as ERR 09.

GENERAL MESSAGES

ERR 13

Meaning: BG segment precedes BG main disc-resident program.

Action: Irrecoverable.

ERR 14

Meaning: Invalid background bounds or illegal response to CHANGE FWA SYS MEM? or to CHANGE BP LINKAGE?

Action: Message is repeated. Enter valid reply.

ERR 15

Meaning: Type 6 program illegally calling a program that is not type 0 or 6.

Action: Revise type 6 program.

ERR 16

Meaning: Base page linkage overflow into system communication area.

Action: Diagnostic printed for each word required (communication area is used). Revise order of program loading or CHANGE BP LINKAGE answers to reduce linkage requirements.

ERR 17

Meaning: Current disc address exceeds number of available tracks.

Action: Irrecoverable error.

ERR 18

Meaning: Memory overflow (absolute code exceeds LWA memory).

Action: Diagnostic printed for each word required (absolute code is generated beyond LWA). Revise program or BG BOUNDARY answer.

ERR 19

Meaning: Restart of RTGEN attempted during final clean up operations.

Action: Irrecoverable error.

ERR 20

Not used

ERR 21

Meaning: Module containing entry point $CIC not loaded.

Action: Irrecoverable error.

ERR 22

Meaning: Read parity/decode disc error. A-Register bits 7-14 show track number; bits 0-6 show sector number.

Action: After ten attempts to read or write the disc sector, the computer halts. To try ten more times, press RUN.

ERR 23

Meaning: Invalid FWA BP LINKAGE.

Action: Message repeated; enter valid reply.

MESSAGES DURING I/O TABLE ENTRY

ERR 24

Meaning: Invalid channel number.
Action: Enter valid EQT statement.

ERR 25

Meaning: Invalid driver name or no driver entry points.

Action: Same as ERR 24.

ERR 26

Meaning: Invalid or duplicate D, B, T operands.

Action: Same as ERR 24.

ERR 27

Meaning: Invalid logical unit number.

Action: Enter valid DRT statement.

ERR 28

Meaning: Invalid channel number.

Action: Enter valid INT statement.

ERR 29

Meaning: Channel number decreasing.

Action: Same as ERR 28.

ERR 30

Meaning: Invalid mnemonic.

Action: Same as ERR 28.

ERR 31

Meaning: Invalid EQT number.

Action: Same as ERR 28.

ERR 32

Meaning: Invalid program name.

Action: Same as ERR 28.

ERR 33

Meaning: Invalid entry point.

Action: Same as ERR 28.

ERR 34

Meaning: Invalid absolute value.

Action: Same as ERR 28.

ERR 35

Meaning: Base page interrupt locations overflow into linkage area.

Action: Re-start Disc Loading Phase at FWA BP LINKAGE? request.

ERR 36

Meaning: Invalid number of characters in final operand.

Action: Same as ERR 28.

GENERAL MESSAGE

ERR 37
name

Meaning: Invalid declaration of common in system or library programs (*name* is the illegal program).

Action: Revise the program.

ERR 38

Meaning: System area overflows scratch area. (This error sometimes possible when restarting disc loading phase; irrecoverable.)

Action: Check order of loading programs or move scratch boundary up.

ERR 39
name

Meaning: System illegally referenced a type 6 program (*name* is the type 6 program).

Action: Revise the program.

ERR 40

Meaning: First system track defective or first scratch track defective.

Action: Redefine the track areas.

ERR 41

Meaning: More than 10 bad tracks on system, auxiliary and scratch discs combined.

Action: Redefine track area.

ERR 42

Meaning: Absolute system area contains a bad track.

Action: Redefine track area.

ERR 43

Meaning: Disc specifications do not conform to system disc, or not enough spare tracks are available.

Action: Redefine track area or sectors per track answers.

PART 3
Fixed Head System Generation

INTRODUCTION

The set-up and operation of a Fixed Head RTE-II System involves two essential steps and one optional step; the system must be configured using the Real-Time System Generator, RTGEN (HP Part No. 92001-16xxx); it must be initiated from the disc by the core-resident basic binary disc loader (BBDL); and it may be dumped onto tape using the System Dump, SDUMP, as protection against a disc failure.

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

FIXED HEAD DISC RTGEN

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

To accomplish this, RTGEN requests certain information from the user; then it accepts the relocatable program modules to be included in the system, determines where they belong in the core, relocates them into absolute format, and stores them on the disc. RTGEN also creates I/O tables by identifying each I/O device and its associated driver routine, and establishing procedures for interrupt processing on each channel. All of this information is pre-planned and is located on the Configuration Worksheets filled out in Part 1. The answers used in this part are from an example RTE-II System that has been configured on worksheets per the instructions in Part 1. The completed worksheets and RTGEN listing are located in Appendix C.

RTGEN is an absolute program, loaded into core by the BBDL from paper tape. Since RTGEN is independent of the system which it generates, the I/O operations of RTGEN require SIO Drivers for the devices used in generation.

OPERATING PROCEDURES

The operation of RTGEN involves five phases:

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

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

c. PARAMETER INPUT PHASE. Parameters describing or changing the type, priority, and execution interval of each program may be entered. (Although this information may already be included in the program's NAM record, it can be changed at this point.) Entry points are changed and a number of blank ID segments are reserved for subsequent program addition. RTGEN requests a specification of the base page linkage and begins loading programs onto the disc.

d. TABLE GENERATION PHASE. RTGEN requests information about Class I/O, LU switching, Resource Numbers, and buffer limits. RTGEN then requests information for the equipment table, device reference table, and interrupt table and finishes loading the rest of the programs onto the disc.

e. SYSTEM BOUNDARIES PHASE. RTGEN begins loading the system and stops at each boundary and reports the address. If the user desires, the address can be moved up to a page boundary. The gap between the two then reverts to system use.

NOTE

During the first three phases, RTGEN can be restarted at octal address 100. The restarting points are back to the beginning of the Initialization Phase, Program Input Phase, or Parameter Input Phase.

Make certain that the switch register options are set up before pressing RUN.

To execute RTGEN and configure an RTE-II System, follow these steps:

a. Turn on all equipment; set the system teleprinter to LINE, and disable the disc protect switch on the interface card. (See Drum Memory Interface Kit Manual, or Disc Memory Interface Kit Manual.)

b. System Input/Output (SIO) drivers for the various devices to be used during the generation must be loaded (or configured into one tape). Use the following list as a guide.

Teletype	– required.
Line printer	– optional. If used, all teletype output goes to the line printer
Photoreader	– optional. For program/command input.
Punch	– optional. For punching bootstrap.
Mag Tape	– optional. For program input.
Disc	– optional. For program input.

c. Load the RTGEN tape using the BBDL.

d. If a fixed head disc is to be used to store the relocatable modules, its SIO driver must be configured as described in the PTS manual on the SIO Disc/Drum Driver Configuration Flowchart.

e. Go to the starting address of the program, octal 100. Before RUN is pressed, refer to Table 6-7 for any switch register options.

INITIALIZATION PHASE

Editor's note – An example fixed head generation was not available at press time. The shaded responses are for reference only and do not reflect an operating system.

During the initialization phase, RTGEN requests information necessary to begin generating the system. After each question is printed, the operator responds with the required answer, followed by carriage return/linefeed (CR/LF). Operator responses are shaded and are only examples; actual responses should be appropriate to the particular system being generated. If an error is made and discovered before LF is entered, type the RUB OUT key then CR/LF. Otherwise restart at step "e" above.

RTGEN requests the higher priority select code (octal) of the system disc or drum unit controller.

FH DISC CHNL?
23

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

SYS DISC SIZE?
32

RTGEN requests the track number starting the disc scratch area. This is the area into which the relocatable software modules will be first loaded by RTGEN during the program input phase. If there is a large quantity of relocatable modules the scratch area needs to be large enough to accommodate them all; however, there is also a limit to how small the number can be. The absolute system is started on track zero (0), and built upwards toward and sometimes into the relocatable modules overlaying the latter after they have been converted into absolute code. Therefore, the scratch area must be started at a point high enough so that the absolute system will not overlay the relocatable modules before they are used.

START SCRATCH?
4 (hp)

RTGEN requests the number of software protected tracks.

NO. PROTECTED?
8

RTGEN requests the number of 64-word sectors (decimal) per track on the system disc or drum.

#SECTORS/TRACK?
128

RTGEN requests the number of tracks on the auxiliary disc or drum.

AUX DISC SIZE?

Operator responds with number of tracks (or zero (0) if there is no auxiliary disc).

0

(hp) *An entry of 0 defaults the start scratch to the midpoint of the available disc space (e.g., if disc size is 32, a 0 entry starts the scratch at track 16.)*

NOTE

If an auxiliary disc is specified (moving head or fixed head), the number of sectors per track question relates to the number of 64-word sectors per track. In the above case, there is no auxiliary disc specified so the # SECTORS/ TRACK question is not asked.

RTGEN requests the select code of the time base generator (octal).

TBG CHNL?
12

RTGEN requests the address of the privileged interrupt I/O card (if present).

PRIV. INT. CARD ADDR?

Operator responds with the octal select code of the privileged interrupt HP 12620 card (and all devices in higher priority slot become privileged) or zero if the card is not used.

11

RTGEN asks if swapping is allowed in the foreground area. Answer YES or NO.

FG SWAPPING?
YES

RTGEN asks if swapping is allowed in the background area. Answer YES or NO.

BG SWAPPING?
YES

RTGEN asks if any program is allowed to be locked into the foreground. Answer YES or NO.

FG CORE LOCK?
YES

RTGEN asks if any program is allowed to be locked into the background. Answer YES or NO.

BG CORE LOCK?
YES

RTGEN next requests the amount of swap delay time. Enter a decimal number between 0 and 255 (meaning tens of milliseconds).

SWAP DELAY?
25

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

LWA MEM?

Operator responds with 37677 for 16K, 57677 for 24K, or 77677 for 32K.

37677

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

PRGM INPT?

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

PT

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

LIBR INPT?

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

PT

NOTE

Any type of program can be entered through the program input unit or the library input unit.

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

PRAM INPT?

Operator responds with PT or TY.

TY

When RTGEN finishes the initialization phase, the hardware disc protect switch on the disc controller is checked. If the disc is protected RTGEN prints

TURN OFF DISC PROTECT – PRESS RUN

and halts with the Memory Data Register = 102032. The operator must turn off the disc protect switch and press RUN to continue to the next phase. If the switch already is off, the message is not printed and the computer halts with the Memory Data Register = 102077. If the response to PGM INPT or LIB INPT was MT, the magnetic tape unit will rewind to the load point, and then space forward to relocatable file number two, before the HALT 102077. A HALT 102044 indicates that the magnetic tape unit is not ready. Set it up and press RUN.

PROGRAM INPUT PHASE

During the program input phase, RTGEN accepts relocatable programs from the program input unit and library input unit specified during the initialization phase. The operator selects the input device by setting switch register bits 0-1.

00 = program input unit
10 = library input unit
01 = print list of undefined externals, or after the printout

01 = terminate input phase

If an error is detected during the Program Input Phase, the name of the offending program is printed. At this point the operator can set or clear bit 15 before pushing RUN to accomplish one of the following:

Bit 15 = 1 The program just read will be purged. Input data is ignored until the next NAM record.

Bit 15 = 0 The program just read is not purged. The last record can be re-read.

Relocatable programs should be loaded in the following order. Note that some of the programs may not be present in some configurations.

Core Resident System
I/O Drivers
Power Fail (DVP 43)
System Programs written by the user
Multi-Terminal Monitor
Foreground Core-Resident Programs
Foreground Disc-Resident Programs
Background Core-Resident Programs
Assembler (Main and its Segments)
FORTRAN (Main and its Segments) and/or
FORTRAN IV (Main and its Segments) but not both FORTRAN IV Versions
ALGOL
Auto Restart
Relocating Loader
Editors
Batch Monitor
Other Background Disc-Resident Programs and their respective segments, if any.
System Library
Batch Monitor Library
Library Programs
Utility Programs

NOTE

If a program is being loaded from paper tape, and was not generated with the type code in the NAM record, the operator can modify it during the Parameter Input Phase.

The operator presses RUN. After a program is loaded, the message "*EOT" is printed whenever an end-of-record occurs. The computer halts.

At this point, the operator has several alternatives:

a. Additional programs can be loaded through the same device by pushing RUN.
b. Input can be switched to the other input device by setting the switch register bits to binary 00 or 10.
c. After each *EOT message, a list of all undefined externals can be printed by setting the switch register bits to binary 01 and pushing RUN. At this point the operator can reset the switches to point to the desired input device, and load additional routines needed to satisfy any undefined externals. If there are none, the message NO UNDEF EXTS is printed and the computer executes a HALT 77. To continue loading programs, reset the switch register to binary 00, or 10, place the program in the input device, and push RUN.
d. To terminate the program input phase, set the switch register to binary 01 and push RUN. (RUN must be pushed again after NO UNDEF EXTS is printed.) If magnetic tape is used for the program input, it will rewind and go off-line at this point (after the second RUN).
e. To restart this phase go to the starting address of the program, octal 100, clear the switch register, and push RUN.

PARAMETER INPUT PHASE

If the teletype was not specified as the PRAM INPT device during initialization, the computer executes a HALT 77 to wait for the parameter tape to be inserted in the photo-reader. Push RUN to continue. If there are any errors on the parameter tape, they will be printed on the list device.

During the parameter input phase, the operator can modify the type, priority, or execution intervals (in decimal) of any of the programs entered during the program input phase (except that the program type code of background main programs and their segments cannot be changed without losing their relationship to each other).

Each parameter record is of this general form:

name, type [,*priority*] [,*execution interval*]

Refer to the Configuration Worksheet and enter any modification parameters that are listed. If there are none, enter a /E on the teletype. As an example (see Appendix C) the operator can respond with:

PARAMETERS

```
#BSC,14
WHZAT,2,10
MEM,1,32767
SYSON,82,32766
RNRQ,14
LURQ,14
EQLU,14
/E
```

NOTE

Refer to Part 1, Parameter Input Phase for an explanation of type codes 81 through 92.

RTGEN requests if there are any entry records to be changed. The numbers are assumed octal unless followed by a "D" which signifies decimal.

CHANGE ENTS?

```
*EAU MACRO'S
.MPY,RP,100200
.DIV,RP,100400
.DLU,RP,104200
.DST,RP,104400
*HFP MACRO'S
.FAD,RP,105000
.FSB,RP,105020
.FMP,RP,105040
.FDV,RP,105060
IFIX,RP,105100
FLOAT,RP,105120
*FFP MACRO'S
DBLE,RP,105210
SNGL,RP,105202
.XMPY,RP,105203
.XDIV,RP,105204
.DFER,RP,105205
.XADD,RP,105213
.XSUB,RP,105214
.GOTO,RP,105221
..MAP,RP,105222
.ENTR,RP,105223
.ENTP,RP,105224
/E
```

RTGEN requests the number of blank 28-word ID segments to be allocated for on-line loading of programs by the relocating loader. The operator responds with a one or two digit decimal number (zero is changed to one, because one is required to do any on-line loading); 29 words are reserved in the resident table area for each blank ID segment.

OF BLANK ID SEGMENTS?
10

RTGEN next requests the number of blank 9-word ID segments to be allocated for background segments. The operator responds with a one or two digit decimal number (zero is legal); 10 words are reserved in the resident table area for each blank BG ID segment.

OF BLANK BG SEG. ID SEGMENTS?
25

Before the next question concerning Base Page Linkages is answered, switch 14 should be set or cleared according to the following directions. With bit 14 set, RTGEN, as each module is loaded, creates at each end of a module which crosses a page boundary, a current page-link buffer. The current page-link buffer is used for DBL records that reference addresses across the current page boundary. If this current page-link buffer overflows, then RTGEN reverts to using base page links for the module. The end result is that base page is saved as links are established in the current page buffer.

NOTE

EXT references and Assembly Language type 3 or 5 programs still

use exclusively base page for establishing links. However, this is not true for subroutines referenced by these programs.

RTGEN requests the first word of available core memory in base page. Operator responds with the first available octal select code number after the last I/O card.

FWA BP LINKAGE?
100

Disc loading begins with the modules of the system, including I/O drivers. As RTGEN loads these programs, it prints SYSTEM, followed by a memory map giving the starting locations and, if switch register bit 15 is set, the entry points for all main programs and subroutines (subroutines are indented two spaces). Also, if bit 13 is set, the base page linkage is reported after each module is loaded.

TABLE GENERATION PHASE

After the last system module is loaded RTGEN requests how many Class I/O numbers are to be allocated.

*# OF I/O CLASSES?
16

RTGEN next requests the maximum number of logical unit numbers referred to in a single job within the Batch and Spool Monitor.

*# OF LU MAPPINGS?
8

RTGEN next requests how many Resource Numbers are to be allocated.

*# OF RESOURCE NUMBERS?
32

RTGEN next requests the buffer limits.

BUFFER LIMITS (LOW, HIGH)?
100,400

Next, RTGEN generates the three I/O tables; Equipment table, device reference table, and the interrupt table.

RTGEN requests the equipment table entries.

*EQUIPMENT TABLE ENTRY

Operator responds with a series of EQT entries, which are assigned EQT numbers sequentially from one as they are entered. The EQT entry relates the EQT number to an I/O channel and driver. Refer to the Configuration Worksheet and enter the Equipment table entries. The Appendix C example is entered as follows:

Editor's note – entries for this part were unavailable at press time. Similar entries from the moving head part could be referred to as examples.

RTGEN requests the logical unit assignments for the device reference table.

*DEVICE REFERENCE TABLE

For each logical unit number, RTGEN prints

n=EQT#?

where *n* is a decimal integer starting with one.

Operator responds with an EQT entry number appropriate to the standard definition of *n*, and the subchannel number if appropriate. Logical unit numbers 1 through 6 are predefined in the system as:

0 – bit bucket (no action required)
1 – system teleprinter
2 – system mass storage
3 – auxiliary mass storage
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:

Editor's note – entries for this part were unavailable at press time. Similar entries from the moving head part could be referred to as examples.

RTGEN requests the interrupt table entries.

*INTERRUPT TABLE

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 is entered as follows:

Editor's note – entries for this part were unavailable at press time. Similar entries from the moving head part could be referred to as examples.

SYSTEM BOUNDARIES PHASE

When the interrupt table is completed RTGEN prints the new first word of available core memory in base page.

BP LINKAGE 00250

RTGEN reports the starting address of the library and waits for any change. Enter 0 for no change.

LIB ADDRS 34761
CHANGE LIB ADDRS?
36000

RTGEN prints LIBRARY, followed by names (and if bit 15 is set, entry point addresses) of all routines which are referenced by real-time and background programs. After this RTGEN reports the new first word of available core memory in base page.

BP LINKAGE 00431

RTGEN reports the total number of words allocated to foreground common and waits for any change. Enter 0 for no change.

FG COMMON 00000
CHANGE FG COMMON?
200

RTGEN reports the starting address of the foreground core resident area and waits for any change. Enter 0 for no change.

FG RES ADD 44013
CHANGE FG RES ADD?
0

At this time, if any type 9 (foreground core-resident program that uses background common) or type 10 (foreground disc-resident program that uses background common) programs are in the system, RTGEN requests the background boundary. This is the start of background common. Refer to Part 1 for guidelines in determining this entry.

BG BOUNDRY 53600
CHANGE BG BOUNDRY?
0

Next RTGEN prints FG RESIDENTS, and loads the real-time core resident programs and reports the new first word of available core memory in base page.

BP LINKAGE 00460

NOTE

If there are no library or real-time core resident programs, RTGEN prints (NONE) and does not report the links because they have not changed.

RTGEN reports the starting address of foreground disc resident area and waits for any change. Enter 0 for no change.

FG DSC ADD 45600
CHANGE FG DSC ADD?
46000

RTGEN prints FG DISC RESIDENTS, loads the real-time disc-resident programs, reports the new first word of available core memory in base page, and waits for any change. Increasing the base page linkage area here allows for future additions of larger real-time disc-resident programs on-line with the RTE-II Relocating Loader.

BP LINKAGE 00631
CHANGE BP LINKAGE?

Operator responds with zero for no change, or an octal address greater than the last base page link reported. Refer to Part 1 for guidelines in determining this entry. The Appendix C entry is:

740

RTGEN next reports the first word address of the system available memory and waits for a change. Enter a 0 for no change or an octal address greater than the number reported. Increasing this number allows more room for on-line additions with the Loader.

SYS AVMEM 53527
CHANGE SYS AVMEM?
53600

Next, if there are no type 9 or 10 programs in the system, RTGEN reports the first address of the background area. This establishes the area between the SYS AVMEM and the background and is used for temporary storage of output buffers, and re-entrant temporaries; it should be large enough to handle the largest anticipated I/O transfer. Refer

to Part 1 for guidelines in determining this entry. If there are type 9 or 10 programs in the system, this question was asked earlier and is not repeated here.

```
BG BOUNDRY 53600
CHANGE BG BOUNDRY?
 0
```

RTGEN reports the total number of words allocated to background common and waits for any change. Enter 0 for no change.

```
BG COMMON 00000
CHANGE BG COMMON?
 200
```

RTGEN reports the new background common size (unless it is zero).

```
BG COM  00200
```

RTGEN reports the starting address of the background core-resident area and waits for any change. Enter 0 for no change.

```
BG RES ADD 54000
CHANGE BG RES ADD?
 0
```

Next RTGEN prints BG RESIDENTS, and loads the background resident programs. If there are no background core-resident programs RTGEN prints (NONE) and does not report the base page links because they have not changed.

RTGEN reports the starting address of the background disc-resident area and waits for any change. Enter 0 for no change.

```
BG DSC ADD 54000
CHANGE BG DSC ADD?
 0
```

RTGEN prints BG DISC RESIDENTS, proceeds to load the background disc-resident programs, and prints names and entry points for mains and subroutines. RTGEN completes the loading and gives the total base page links used.

```
BP LINKAGE 01427
```

RTGEN then reports the system is loaded on the disc by printing

```
SYSTEM STORED ON DISC
```

RTGEN gives the address of the last sector and track used in storing the system on the disc, and halts.

```
LAST SYS DISC ADDR: TRK 025 SEC 032 (10)
```

where both the track number and the sector number are decimal numbers. These should be recorded if the system is going to be dumped on to tape for back-up protection using SDUMP.

INITIATING RTE-II FROM THE FIXED HEAD DISC

The basic binary disc loader (BBDL) is used to load the system into core as follows:

a. The operator goes to the starting octal address of the BBDL;

037760 for 16K
057760 for 24K
077760 for 32K

and pushes RUN.

b. When the computer halts with 102077 in the memory data register, the operator clears the switch register and presses RUN. When RTE-II is loaded, it will type:

```
SET TIME
```

The operator either sets the clock to current time using the TM operator request, or enters any other request (the system starts at approximately 8:00 a.m. on the release date).

SDUMP

SDUMP, the System Dump, is an independent absolute program that can create back-up copies of fixed head disc-based systems on punched tape or magnetic tape. The back-up copy can later be reloaded onto the fixed head disc by SDUMP. In a fixed head system with a moving head auxiliary disc, SDUMP will only dump that portion of the system located on the fixed head disc.

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

After loading SDUMP, go to starting address octal 100, and press RUN. SDUMP prints out a request guide on teleprinter:

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

Then, SDUMP requests the lower-number select code of the disc in octal.

```
DISC CHNL?
 23
```

After SDUMP types

```
COMMAND
```

Operator responds with *V, L, T,* or *D* (*D* requires parameters).

```
D,0,10-7
```

An explanation of the parameters that SDUMP uses is as follows:

D for dumping requires octal parameters specifying the track or tracks to be dumped. For example, the operator response above tells SDUMP to dump the absolute binary code stored on tracks 0 through 10 up to and including sector 7. If only one number is entered, only that track is dumped. These numbers are obtained from the RTGEN listing. RTGEN gives the address of the last sector and tracks used in storing the RTE system under the heading

```
LAST SYS DISC ADDR:
  TRK nn SECmmm(10)
```

It should be noted that RTGEN gives the track and sector numbers in decimal. Use Table 6-11 to convert these numbers to octal for SDUMP. (First physical track of the disc is decimal 0.)

The sector parameter (*-S*) is optional. If output is to paper tape, trailer and leader blank tape is produced, and two tracks are dumped at a time. Always start with a new roll of tape. If output is to magnetic tape, the entire information is dumped, followed by an End-Of-File which is written over by subsequent dumps.

Table 6-11. Octal/Decimal Conversion

Octal	Decimal	Octal	Decimal
0	0	16	14
1	1	17	15
2	2	20	16
3	3	21	17
4	4	22	18
5	5	23	19
6	6	24	20
7	7	25	21
10	8	26	22
11	9	27	23
12	10	30	24
13	11	31	25
14	12	32	26
15	13	33	27

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

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

T by itself is for termination. However, *T* used in the parameter explanation stands for track.

An illegal command causes the message:

STATEMENT ERROR

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

PARAM ERROR: NON-NUMERIC OR NON-OCTAL

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

SDUMP ERROR MESSAGES

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

STATEMENT ERROR

Re-type input statement incorrect format.

EOT

The end of the input tape being read has been reached; either load the next tape or go on to the next phase.

CHANGE OUTPUT TAPE, HIT RUN

Two full tracks have been dumped onto paper tape; perform the requested action.

TURN OFF DISC PROTECT, HIT RUN

Set the disc track protect switch off, then press RUN.

DISC INPUT ERROR

Disc error diagnostic, for a parity or decode or abort status after 10 re-trys. Input sequence repeated on restart.

DISC WRITE ABORT

Disc error diagnostic, for an abort status after a write attempt. Sequence is repeated if restarted.

TRACK 27(8) SECTOR 107(8)

Identification information for the disc and tape error diagnostics are described as follows:

TAPE/DISC VERIFY ERROR

Disc and tape records do not agree. Disc record is rewritten on restart.

TAPE CHECKSUM ERROR

The checksum in the tape record does not match the sum computed by SDUMP. Current record is ignored if restarted.

MT ERROR – READ PARITY
MT ERROR – EOT RESTART

Magnetic tape errors. Error recovery procedures are completed by driver. Restart to re-try sequence.

FH RTGEN ERROR MESSAGES

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

MESSAGES DURING INITIALIZATION AND INPUT PHASE

ERR 01

Meaning: Invalid response to initialization request.

Action: Message is repeated. Enter valid reply.

ERR 02

Meaning: Checksum error on program input.

Action: Computer halts; reposition tape to beginning of record and press RUN to reread. If input is from MT or DF, it is automatically backspaced when RUN is pressed unless bit 15 is set.

ERR 03

Meaning: Record out of sequence.

Action: Same as ERR 02.

ERR 04

Meaning: Illegal record type.

Action: Same as ERR 02.

ERR 05
name

Meaning: Duplicate entry point.

Action: Revise program by re-labeling the entry points (the current entry point replaces the previous entry point).

ERR 06

Not used

ERR 07

Meaning: Program name or entry point table overflow of available memory.

Action: Irrecoverable error. Revise or delete programs.

ERR 08
name

Meaning: Duplicate program name.

Action: The current program replaces the previous program.

MESSAGES DURING THE PARAMETER PHASE

ERR 09

Meaning: Parameter name error (no such program).

Action: Enter valid parameter statement.

ERR 10

Meaning: Parameter type error.

Action: Same as ERR 09.

ERR 11

Meaning: Parameter type error.

Action: Same as ERR 09.

ERR 12

Meaning: Execution interval error.

Action: Same as ERR 09.

GENERAL MESSAGES

ERR 13

Meaning: BG segment precedes BG main disc-resident program.

Action: Irrecoverable.

ERR 14

Meaning: Invalid background bounds or illegal response to CHANGE FWA SYS MEM or to CHANGE BP LINKAGE?

Action: Message is repeated. Enter valid reply.

ERR 15

Meaning: Type 6 program illegally calling a program that is not type 0 or 6.

Action: Revise type 6 program.

ERR 16

Meaning: Base page linkage overflow into system communication area.

Action: Diagnostic printed for each word required (communication area is used). Revise order of program loading or CHANGE BP LINKAGE answer to reduce linkage requirements.

ERR 17

Meaning: Current disc address exceeds number of available tracks.

Action: Irrecoverable error.

ERR 18

Meaning: Memory overflow (absolute code exceeds LWA memory).

Action: Diagnostic printed for each word required (absolute code is generated beyond LWA). Revise program or BG BOUNDARY answer.

ERR 19

Not used

ERR 20

Not used

ERR 21

Meaning: Module containing entry point $CIC not loaded.

Action: Irrecoverable error.

ERR 22

Meaning: Read parity/decode disc error. A-Register bits 7-14 show track number; bits 0-6 show sector number.

Action: After ten attempts to read or write the disc sector, the computer halts. To try ten more times, press RUN.

ERR 23

Meaning: Invalid FWA BP LINKAGE

Action: Message repeated; enter valid reply.

MESSAGES DURING I/O TABLE ENTRY

ERR 24

Meaning: Invalid channel number.

Action: Enter valid EQT statement.

ERR 25

Meaning: Invalid driver name or no driver entry points.

Action: Same as ERR 24.

ERR 26

Meaning: Invalid or duplicate D, B, T operands.

Action: Same as ERR 24.

ERR 27

Meaning: Invalid logical unit number.

Action: Enter valid DRT statement.

ERR 28

Meaning: Invalid channel number.

Action: Enter valid INT statement.

ERR 29

Meaning: Channel number decreasing.

Action: Same as ERR 28.

ERR 30

Meaning: Invalid mnemonic.

Action: Same as ERR 28.

ERR 31

Meaning: Invalid EQT number.

Action: Same as ERR 28.

ERR 32

Meaning: Invalid program name.

Action: Same as ERR 28.

ERR 33

Meaning: Invalid entry point.

Action: Same as ERR 28.

ERR 34

Meaning: Invalid absolute value.

Action: Same as ERR 28.

ERR 35

Meaning: Base page interrupt locations overflow into linkage area.

Action: Re-start Disc Loading Phase at FWA BP LINKAGE? request.

ERR 36

Meaning: Invalid number of characters in final operand.

Action: Same as ERR 28.

GENERAL MESSAGE

ERR 37
name

Meaning: Invalid declaration of common in system or library programs (*name* is the illegal program).

Action: Revise the program.

ERR 38

Meaning: System area overflows scratch area. (This error sometimes possible when restarting disc loading phase; irrecoverable).

Action: Check order of loading programs or move scratch boundary up.

ERR 39
name

Meaning: System illegally referenced to a type 6 program (*name* is the type 6 program).

Action: Revise the program.

APPENDIX A
SYSTEM TABLES AND COMMUNICATION AREA

BASE PAGE COMMUNICATION AREA

A block of storage in base page, starting at octal location 1650, contains the system communication area and is used by RTE-II to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, etc. The Real-Time Assembler allows absolute references into this area (i.e., less than 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.

Octal Location	Contents	Description
SYSTEM TABLE DEFINITION		
01650	EQTA	FWA of equipment table
01651	EQT#	No. of EQT entries
01652	DRT	FWA of device reference table
01653	LUMAX	No. of logical units (in DRT)
01654	INTBA	FWA of interrupt table
01655	INTLG	No. of interrupt table entries
01656	TAT	FWA of track assignment table
01657	KEYWD	FWA of keyword block
I/O MODULE/DRIVER COMMUNICATION		
01660	EQT1	Addresses of first 11-words of current EQT entry (see 01771 for last 4 words)
01661	EQT2	
01662	EQT3	
01663	EQT4	
01664	EQT5	
01665	EQT6	
01666	EQT7	
01667	EQT8	
01670	EQT9	
01671	EQT10	
01672	EQT11	
01673	CHAN	Current DMA channel No.
01674	TBG	I/O address of time-base card
01675	SYSTY	EQT entry address of system TTY
SYSTEM REQUEST PROCESSOR/'EXEC' COMMUNICATION		
01676	RQCNT	No. of request parameters - 1
01677	RQRTN	Return point address
01700	RQP1	Addresses of request parameters (set for maximum of 8 parameters)
01701	RQP2	
01702	RQP3	
01703	RQP4	
01704	RQP5	
01705	RQP6	
01706	RQP7	
01707	RQP8	
ADDRESSES OF SYSTEM LISTS		
01711	SKEDD	'Schedule' list
01713	SUSP2	'Wait suspend' list
01714	SUSP3	'Available memory' list
01715	SUSP4	'Disc allocation' list
01716	SUSP5	'Operator suspend' list
DEFINITION OF EXECUTING PROGRAM ID SEGMENT		
01717	XEQT	ID segment addr. of current program
01720	XLINK	'Linkage'
01721	XTEMP	'Temporary' (5-words)
01726	XPRIO	'Priority' word
01727	XPENT	'Primary entry point'
01730	XSUSP	'point of suspension'
01731	XA	'A-Register' at suspension
01732	XB	'B-Register' at suspension
01733	XEO	'E and overflow' at suspension
SYSTEM MODULE COMMUNICATION FLAGS		
01734	OPATN	Operator/keyboard attention flag
01735	OPFLG	Operator communication flag
01736	SWAP	RT disc resident swapping flag
01737	DUMMY	I/O address of dummy int. card
01740	IDSDA	Disc addr. of first ID segment
01741	IDSDP	Position within sector

Octal Location	Contents	Description
DEFINITION OF MEMORY ALLOCATION BASES		
01742	BPA1	FWA RT disc res. BP link area
01743	BPA2	LWA RT disc res. BP link area
01744	BPA3	FWA bkg disc res. BP link area
01745	LBORG	FWA of resident library area
01746	RTORG	GWA of RT area
01747	RTCOM	Length of RT common area
01750	RTDRA	FWA of RT disc resident area
01751	AVMEM	FWA of system available memory
01752	BKGRG	FWA of background area
01753	BDCOM	Length of background common area
01754	BKDRA	FWA of DKG disc resident area
UTILITY PARAMETERS		
01755	TATLG	Length of track assignment table
01756	TATSD	# of tracks on system disc
01757	SECT2	# sectors/track on LU 2 (system)
01760	SECT3	# sectors/track on LU 3 (aux.)
01761	DSCLB	Disc addr of res lib entry pts
01762	DSCLN	# of res lib entry points
01763	DSCUT	Disc addr of reloc utility programs
01764	DSCUN	# of reloc utility progs
01765	LGOTK	Load-n-go; LU, stg track, # of tracks
01766	LGOC	Current LGO track/sector address
01767	SFCUN	Source file LU and disc address
01770	MPTFL	Memory protect on/off flag (0/1)
01771	EQT12	Address of last 4 words of current EQT
01772	EQT13	
01773	EQT14	
01774	EQT15	
01775	FENCE	memory protect fence address
01777	BKLWA	LWA of memory in background

PROGRAM ID SEGMENT

Each user program has a 28 word ID segment located in the system area. The format of the ID segment is shown in Table A-1. The address of each ID segment is located in the Keyword Table (see location 01657). The ID segment contains static and dynamic information defining the properties of a program. The static information is set during generation 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 generation 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.

Table A-1. ID Segment Map

WORD	CONTENTS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	LIST LINKAGE															
2-6	5 WORD TEMPORARY AREA USED FOR SPECIAL FLAGS IN QUEUES (ETC)															
● 7	PRIORITY															
● 8	PRIMARY ENTRY POINT															
9	POINT OF SUSPENSION (XSUSP)															
10	A REGISTER AT SUSPENSION (XA)															
11	B REGISTER AT SUSPENSION (XB)															
12	E/O REGISTERS AT SUSPENSION (XEO)															
● 13	NAME (FIRST AND SECOND CHARACTERS)															
● 14	NAME (THIRD AND FOURTH CHARACTERS)															
● 15	NAME (FIFTH CHARACATER)										TM	CL	AM	SS	TYPE	
16	NA	*	NP	W	A	*	O	*	R	D	*				STATUS	
17	TIME LIST LINKAGE WORD															
● 18	RESOLUTION				T	MULTIPLE										
● 19	LOW ORDER 16 BITS OF EXECUTE TIME LESS 24 HRS. IN 10's MS															
● 20	HIGH ORDER 16 BITS OF EXECUTE TIME															
21	BA	FW	*	AT	RM	RE	PW	RN	FATHER ID-SEG. NUMBER							
22	THIS WORD RESERVED FOR FUTURE IMPROVEMENTS															
● 23	LOW MAIN ADDRESS															
● 24	HI MAIN ADDRESS +1															
● 25	LOW BASE PAGE ADDRESS															
● 26	HI BASE PAGE ADDRESS +1															
● 27	DISC ADDRESS (LU (15,) TRACK (14-7), SECTOR (6-0)															
28	SWAP DISC ADDRESS (LU (15), TRACK (14-7), = TRACKS (6-0)															

PORTION OF ID SEGMENT NORMALLY USED BY MEMORY RESIDENT PROGRAMS. LINK IS STILL TO WORD 1.

● WORDS USED IN SHORT ID SEGMENT

WHERE:

* = These bits are reserved for future improvements.
TM = Temporary load (copy of ID segment is not on the disc).
CL = Core lock (program may not be swapped).
AM = All memory (program uses all of its area).
SS = Short segment (indicates a 9-word ID segment).
NA = No abort (pass abory errors to the program instead).
NP = No parameters allowed on reschedule.
W = Wait bit (waiting for program whose ID segment address is in word 2).
A = Abort on next list entry for this program.
O = Operator suspend on next schedule attempt.
R = Resource save (save resources when setting dormant).
D = Dormant bit (set dormant on next schedule attempt).
T = Time list entry bit (program is in the time list).
BA = Batch (program is running under batch).
FW = Father is waiting (he scheduled with wait).
AT = Attention bit (operator has requested attention).
RM = Re-entrant memory must be moved before dispatching program.
RE = Re-entrant routine in control now.
PW = Program wait (some program wants to schedule this one).
RN = Resource number either owned or locked by this program.

THE EQUIPMENT TABLE

The Equipment Table (EQT) has an entry for each device recognized by RTE-II (these entries are established by the user when the RTE-II System is generated). These 15-word EQT entries reside in the system, and have format as shown in Table A-2.

Table A-2. EQT Table Entries

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	P	S	T	Unit #					Channel					
5	AV		EQ 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															

Where:

D = 1 if DMA required.

B = 1 if automatic output buffering used.

P = 1 if driver is to process power fail.

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.

EQ TYPE CODE = type of device. When this octal number is linked with "DV*x*," it identifies the device's software driver routine as follows:

- 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.
- 15 = mark sense card reader.
- 20 to 37 = magnetic tape/mass storage devices.
- 31 = 7900 moving head disc.
- 32 = 7905 moving head disc.
- 40 to 77 = instruments.

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

DEVICE REFERENCE TABLE

Logical unit numbers 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 the generator, and can be modified by the LU operator request (see Figure A-1).

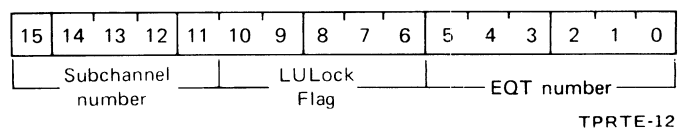


Figure A-1. Device Reference Table Word

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-II 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

DISC LAYOUT OF RTE-II SYSTEM

Figure A-2 diagrams the allocation of disc space by the system generator when it creates an RTE-II System. The bottom portion of the figure shows the difference between loader area in the moving head and fixed head system.

BBDL LISTING

Figure A-3 is an Assembly listing of the Basic Binary Fixed Head Disc Loader, Figure A-4 is a listing of 7900 Moving Head Disc Loader and Figure A-5 is a listing of the 7905 Moving Head Disc Loader. The loader resides in the protected area that is the highest 64 words of core. If the loader is destroyed, it can be replaced through the switch register using this listing. The operator simply replaces symbolic items with the value appropriate to the configuration.

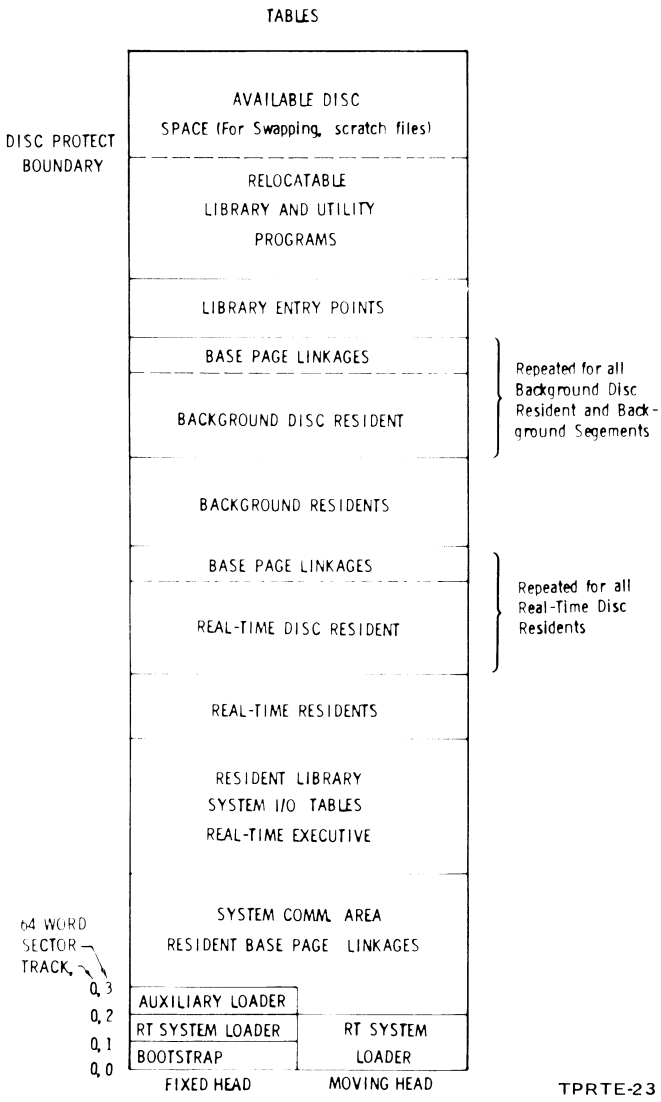


Figure A-2. Disc Space Allocation in RTE-II System

```
PAGE 0002 #01  BASIC BINARY FIXED HEAD DISC LOADER


0001                    ASMB,B,L,A
0003*
0004*
0005*
0006*
0007  L7700                    ORG L7700B
0008*
0009  L7700 107700  START CLC 0,C          TURN OFF ALL DEVICES
0010  L7701 002401        CLA,RSS          BYPASS LEADER
0011  L7702 063726  CONT  LDA CM21         SET A FOR EOT TEST
0012  L7703 006700        CLB,CCE          SET UP TO READ ONLY 1 CHARACTER
0013  L7704 017742        JSB READ1
0014  L7705 007306  LEADR CMB,CCE,INB,SZB  STILL LEADER?
0015  L7706 027713        JMP RECLN        NO,PROCESS RECORD LENGTH
0016  L7707 002006  FOTCH INA,SZA          YES,CHECK FOR EOT
0017  L7710 027703        JMP CONT+1       TRY AGAIN
0018  L7711 102077        HLT 77B          EOT HALT
0019  L7712 027700        JMP START        LOAD NEXT PROGRAM
0020  L7713 077754  RECLN STB COUNT        NUMBER OF INSTRUCTION WORDS
0021  L7714 017742        JSB READ1        BYPASS FEED FRAME
0022  L7715 017742        JSB READ1        GET LOAD ADDRESS
0023  L7716 074000        STB A            INITIAL INPUT FOR CHECKSUM
0024  L7717 077757        STB ADDR
0025  L7720 067757  SUCID LDB ADDR
0026  L7721 047755        ADB MAXAD
0027  L7722 002040        SEZ              TRYING TO LOAD IN LOADER?
0028  L7723 027740        JMP RESCU        YES
0029  L7724 017742  LOAD  JSB READ1        NO,E IS SET FOR READING A WORD
 030  L7725 040001        ADA B            ACCUMULATE CHECKSUM
0031  L7726 177757  CM21  STB ADDR,I       ALSO USED FOR TRAILER COUNT
0032  L7727 037757        ISZ ADDR
0033  L7730 000040        CLE              SET FOR READING A WORD
0034  L7731 037754        ISZ COUNT
0035  L7732 027720        JMP SUCID
0036  L7733 017742        JSB READ1        READ PUNCHED CHECKSUM
0037  L7734 054000        CPB A            COMPARE WITH COMPUTED CHECKSUM
0038  L7735 027702        JMP CONT         CONTINUE
0039  L7736 102011        HLT 11B          CHECKSUM ERROR
0040  L7737 027700        JMP START
0041  L7740 102055  RESCU HLT 55B          TRIED TO LOAD IN LOADER
0042  L7741 027700        JMP START
0043  L7742 077742  READ1 NOP
0044  L7743 006600        CLB,CME          TOGGLE WORD/CHARACTER INDICATOR
0045  L7744 1037PR  READ2 STC PR,C
0046  L7745 1023PR        SFS PR
0047  L7746 027745        JMP *-1
0048  L7747 1074PR        MIB PR,C         DON'T LEAVE PENDING INTERRUPT.
0049  L7750 002041        SEZ,RSS
0050  L7751 127742        JMP READ1,I
0051  L7752 005767        BLF,CLE,BLF,     SET UP FOR NEXT CHARACTER
0052  L7753 027744        JMP READ2
0053  L7754 077754  COUNT NOP
0054  L7755 1M0100  MAXAD ABS -START
0055  L7756 020000  CWORD ABS 20000B+DC     STC DISC AFTER EACH WORD
0056  L7757 077757  ADDR  NOP
0057*
```

Figure A-3. Basic Binary Fixed Head Disc Loader

```
PAGE 0003 #01  BASIC BINARY FIXED HEAD DISC LOADER


0058  L7760 107700  DLDR  CLC 0,C       TURN OFF ALL DEVICES
0059  L7761 063756        LDA CWORD
0060  L7762 102606        OTA DMASC      CONTROL WORD TO DMA
0061  L7763 002700        CLA,CCE
0062  L7764 1026CC        OTA CC
0063  L7765 001500        ERA
0064  L7766 102602        OTA DMACL      SET MEM ADDR REG TO START AT 0
0065  L7767 063777        LDA WRDCT
0066  L7770 102702        STC DMACL      ENABLE WORD COUNT
0067  L7771 102602        OTA DMACL      AND TRANSFER TO DMA
0068  L7772 103706        STC DMASC,C    TURN ON DMA
0069  L7773 1027DC        STC DC         TURN ON DISC
0070  L7774 067776        LDB JMP77      SET UP JMP* FOR USE UNTIL DMA
0071  L7775 074077        STB 77B        TRANSFER IS COMPLETE
0072  L7776 024077  JMP77 JMP 77B        EXIT AND WAIT
0073*
0074  L7777 177700  WRDCT OCT -100
0075  00000PR       PR    EQU PR
0076  00006         DMASC EQU 6
0077  00002         DMACL EQU DMASC-4
0078  000DC         DC    EQU DC
0079  000CC         CC    EQU DC+1
0080  00000         A     EQU 0
0081  00001         B     EQU 1
0082                      END
** NO ERRORS*
```

Figure A-3. Basic Binary Fixed head Disc Loader (continued)

```
 PAGE 0002 #01  BINARY PAPER TAPE AND 7900 DISK LOADER


0001                    ASMB,A,L
0003  L7700                    ORG L77008
0004  000PR             RDR   EQU PR
0005  000DC             DC    EQU DC
0006*************************************************
0007*     S. O. S. - R. T. E. - DOS-M BBDL
0008*     REV. D
0009*************************************************
0010*     SOURCE 25311-80020
0011*     ABS    25311-60027
0012*
0013  L7700 002401  BBDL  CLA,RSS
0014  L7701 063721  EOBLK LDA EOTC       TRAILER LENGTH
0015  L7702 107700  LEADR CLC 0,C        LEAVE THINGS CLEAN ON EXIT
0016  L7703 002307        CCE,INA,SZA,RSS
0017  L7704 102077        HLT 77B        END OF TAPE
0018  L7705 017735        JSB RDCH       READ A CHAR
0019  L7706 007307        CMB,CCE,INB,SZB,RSS   IS IT WORD COUNT?
0020  L7707 027702        JMP LEADR      NO.
0021  L7710 077733        STB COUNT      SAVE WORDCOUNT
0022  L7711 017735        JSB RDCH       THROW AWAY ONE FRAME
0023  L7712 017735        JSB RDCH       READ STARTING ADDRESS
0024  L7713 074000        STB 0          INITIALIZE CHECKSUM
0025  L7714 077747  LOOP  STB ADDR       SET POINTER
0026  L7715 047734        ADB MLWAM      VALIDATE LOAD ADDRESS
0027  L7716 002140        SEZ,CLE        DOES IT CLOBBER LOADER?
0028  L7717 102055  ADDR2 HLT 55B        YES. LOSE.
0029  L7720 017735        JSB RDCH       FETCH DATA WORD
0030  L7721 177747  EOTC  STB ADDR,I     PLANT WORD IN CORE
0031  L7722 040001        ADA B          TALLY CHECKSUM
0032  L7723 067747        LDB ADDR
0033  L7724 006104        CLE,INB
0034  L7725 037733        ISZ COUNT      END OF BLOCK?
0035  L7726 027714        JMP LOOP       NOT YET.
0036  L7727 017735        JSB RDCH       READ CHECKSUM.
0037  L7730 054000        CPB 0          VALID?
0038  L7731 027701        JMP EOBLK      YES. READ NEXT BLOCK.
0039  L7732 102011  ADDR1 HLT 11B        CHECKSUM ERROR.
0040*
0041  L7733 000000  COUNT NOP
0042  L7734 1M0100  MLWAM ABS -BBDL
0043*
0044  L7735 000000  RDCH  NOP            READ FRAME (TWO IF E CLEAR)
0045  L7736 006400        CLB
0046  L7737 1037PR        STC RDR,C      START READER
0047  L7740 1023PR        SFS RDR
0048  L7741 027740  MASK  JMP *-1        (USED BY DISK BOOT)
0049  L7742 1074PR        MIB RDR,C
0050  L7743 002240        SEZ,CME        READ ANOTHER CHAR?
0051  L7744 127735        JMP RDCH,I     NO. EXIT WITH E CLEAR.
0052  L7745 005727        BLF,BLF        SET FOR SECOND CHAR
0053  L7746 027737        JMP RDCH+2
0054*
0055  L7747 000000  ADDR  NOP
0056*
```

Figure A-4. Basic Binary 7900 Moving Head Disc Loader

```
 PAGE 0003 #01  BINARY PAPER TAPE AND 7900 DISK LOADER


0058*     DIFFERENCE FROM DOS-M BBDL: BOOT EXT. CALLED WITH JSB.
0059*   THIS BBDL WILL BOOT DOS-M IF "PRESET" IS PRESSED AT
0060*   THE HLT 77.  THE DOS-M BOOT WILL BOOT S. O. S. IF
0061*   THE SWITCH REGISTER HAS THE DISK CHANNEL AT THE HLT 7.
0062*     LOADS TRACK 0 INTO CORE STARTING AT 2011B.
0063*
0064  00CC           CC     EQU DC+1
0065*
0066  L7750 03S000   SEEKC  OCT 3S000
0067  L7751 067741          LDB MASK
0068  L7752 1066CC          OTB CC          ISSUE READ COMMAND
0069  L7753 1037CC          STC CC,C        START READ TO CLEAR FIRST STATUS
0070  L7754 063750          LDA SEEKC
0071  L7755 1026DC          OTA DC          ISSUE CYLINDER ADDRESS (0)
0072  L7756 1037DC          STC DC,C        TELL CTRL. CYL. ADDRS IS LOADED
0073  L7757 1026CC          OTA CC          ABORT READ, SEND SEEK COMMAND
0074  L7760 1037CC          STC CC,C        START SEEK
0075  L7761 063777          LDA DMACW
0076  L7762 102606          OTA SIX         ISSUE DMA CONTROL WORD
0077  L7763 063732          LDA ADDR1
0078  L7764 102602          OTA TWO         ISSUE START CORE ADDRESS (20118)
0079  L7765 1037DC          STC DC,C        TELL CNTR. HEAD/SECT IS LOADED
0080  L7766 102702          STC TWO         SET FOR WORDCOUNT
0081  L7767 102602          OTA TWO         ISSUE WORDCOUNT (HUGE)
0082  L7770 1066CC          OTB CC          ISSUE READ COMMAND (DRIVE 0)
0083  L7771 1037DC          STC DC,C        PREVENT SPURIOUS DMA TRANS.
0084  L7772 103706          STC SIX,C       START DMA
0085  L7773 1037CC          STC CC,C        START DISK READING
 086  L7774 1023CC          SFS CC          WAIT FOR DISK TRANS (6144 WORDS)
0087  L7775 027774          JMP *-1
0088  L7776 117717          JSB ADDR2,I     DONE; JUMP INTO CODE (2055B,I)
0089*
0090  L7777 1200DC  DMACW   ABS 120000B+DC
0091*
0092  00002         TWO     EQU 2
0093  00006         SIX     EQU 6
0094  00001         B       EQU 1
0095*
0096                        END
** NO ERRORS*
*
*  WHERE:
*   DC = DISC LOW NUMBERED SELECT CODE (DATA CHANNEL)
*   CC = DISC HIGH NUMBERED SELECT CODE (COMMAND CHANNEL)
*   PR = PHOTO READER SELECT CODE
*   M  = 0 FOR 32K
*        2 FOR 24K
*        4 FOR 16K
*        6 FOR  8K
*   S  = 1 FOR SUBCHANNEL 0
*        0 FOR SUBCHANNEL 1
*   L  = 1 FOR  8K
*        3 FOR 16K
*        5 FOR 24K
*        7 FOR 32K
*
*
```

Figure A-4. Basic Binary 7900 Moving Head Disc Loader (continued)

```
PAGE 0002 #01  BINARY PAPER TAPE AND 7905 DISK LOADER


0001                    ASMB,B,L,A
0002  L7700                    ORG L7700B
0003*
0004* PAPER TAPE BOOTSTRAP
0005*
0006*     STARTING ADDRESS - L7700B
0007*
0008  L7700 002401  BEGIN CLA,RSS
0009  L7701 063722  START LDA .EOT           GET HOLE COUNT
0010  L7702 107700  AGIN  CLC 0,C
0011  L7703 002307        CCE,INA,SZA,RSS     END OF TAPE?
0012  L7704 102077        HLT 77B             YES - GOOD HLT
0013  L7705 017735        JSB WORD            GET NEXT CHARACTER
0014  L7706 007307        CMB,CCE,INB,SZB,RSS     IS IT WRD CNT?
0015  L7707 027702        JMP AGIN                NO-GET NEXT CHAR
0016*
0017  L7710 077733  TEST  STB WDCNT          STORE B-REG IN WRD CNT
0018  L7711 017735        JSB WORD         GET NEXT CHAR
0019  L7712 017735        JSB WORD          GET ADDRESS WORD
0020  L7713 074000        STB A             STORE
0021  L7714 077747  LOOP  STB ADDR                ADDRESS WORD
0022*
0023  L7715 047734        ADB LOADR          ADD TO LOADR ADDRESS
0024  L7716 002140        SEZ,CLE            IS IT BEYOND LWAM?
0025  L7717 102055  ADDR2 HLT 55B           YES- SUICIDE LOAD
0026*
0027  L7720 017735        JSB WORD            NO-GET NEXT WORD
0028  L7721 040001        ADA B              ADD TO CHECKSUM
0029  L7722 177747  .EOT  STB ADDR,I         PUT WORD IN CORE
0030  L7723 067747        LDB ADDR
0031  L7724 006104        CLE,INB
0032  L7725 037733        ISZ WDCNT          INCREMENT WORD COUNT
0033  L7726 027714        JMP LOOP          NOT DONE-GET NEXT WORD
0034*
0035  L7727 017735        JSB WORD           GET CHECKSUM WORD
0036  L7730 054000        CPB A             COMPARE TO COMPUTED VALUE
0037  L7731 027701        JMP START           OK-TRY NEXT RECORD
0038  L7732 102011  ADDR1 HLT 11B            CHECKSUM ERROR
0039*
0040  L7733 000000  WDCNT NOP
0041  L7734 177100  LOADR ABS -BEGIN
0042*
0043  L7735 000000  WORD  NOP
0044  L7736 006400        CLR              CLEAR B TO MERGE
0045*
0046  L7737 1037PR  CHAR  STC PR,C            GET CHAR
0047  L7740 103SPR        SFS PR                 FROM PHOTOREADER
0048  L7741 027740        JMP *-1
0049  L7742 107APR        MIB PR,C
0050*
0051  L7743 002240        SEZ,CME          IF 2ND CHAR,
0052  L7744 127735        JMP WORD,I             RETURN
0053  L7745 005727        BLF,BLF          IF 1ST CHAR,
0054  L7746 027737        JMP CHAR               GET 2ND
0055*
0056  L7747 000000  ADDR  NOP
```

Figure A-5. Basic Binary 7905 Moving Head Disc Loader

```
PAGE 0003 #01  BINARY PAPER TAPE AND 7905 DISK LOADER


0057*
0058* DISC BOOTSTRAP  -- LOADS DISC PRE-BOOT PROCESSOR FROM
0059*                     TRACK 0, SECTOR 0, DRIVE 0
0060*
0061*                     HEAD # INPUT FROM SWITCH REG
0062*
0063*   STARTING ADDRESS - L77508
0064*
0065*
0066  L7750 067777         LDB ADDR3        FOR INDIRECT JMP, PLACE HLT
0067  L7751 174001  CNT6   STB B,I          FOR INDIRECT JMP, PLACE HLT
0068  L7752 006004         INB              IN 2056B TO REPROTECT THE
0069  L7753 063732         LDA ADDR1        BMDL IF THE PRE-BOOT IS NOT
0070  L7754 170001         STA B,I          LOADED FROM THE DISC.
0071  L7755 067776         LDB DMACW      GET DMA CONTROL WORD
0072  L7756 106606         OTB 6
0073  L7757 106702         CLC 2
0074  L7760 102602         OTA 2          SEND MEM ADDR
0075  L7761 102702         STC 2          SET DMA FOR WORD CNT
0076  L7762 063751         LDA CNT6
0077  L7763 102602         OTA 2           NEED 128 WRDS,GET EXTRA
0078  L7764 102501         LIA 1          GET SUBCHANNEL
0079  L7765 001027         ALS,ALF        PUT SUBCHAN # TO BITS 6 & 7
0080  L7766 013767         AND MSK
0081  L7767 000160  MSK    OCT 160        <CLE,ALS> EXTRA BIT IGNORED
0082  L7770 1067DC         CLC DC         SET "NEXT WRD IS COMND FLG"
0083  L7771 1036DC         OTA DC,C       SEND COLD LOAD COMND
0084  L7772 103706         STC 6,C        START DMA
0085  L7773 1023DC         SFS DC         WAIT UNTIL COMPLETED
0086  L7774 027773         JMP *-1
0087  L7775 117717         JSB ADDR2,I    START BOOT
0088  L7776 1200DC  DMACW  ABS 120000B+DC
0089  L7777 002055  ADDR3  OCT 2055
0090  000PR          PR     EQU PR
0091  000DC          DC     EQU DC
0092  00000          A      EQU 0
0093  00001          B      EQU 1
0094                        END
**  NO ERRORS*
*
*  WHERE:
*   DC = DISC SELECT CODE
*   PR = PHOTO READER SELECT CODE
*   M  = 0 FOR 32K
*        2 FOR 24K
*        4 FOR 16K
*        6 FOR  8K
*   L  = 1 FOR  8K
*      = 3 FOR 16K
*      = 5 FOR 24K
*      = 7 FOR 32K
*
*
```

Figure A-5. Basic Binary 7905 Moving Head Disc Loader (continued)

APPENDIX B
REAL-TIME DISC USAGE

This appendix covers the following subjects:

Track Configuration
Multiple CPU/7905 Operation
Source Record Format

TRACK CONFIGURATION

The configuration of disc tracks is normally done through the interactive generation process described in Section VI. However, when more than one disc controller is needed, the generator dialogue cannot be used and a track map table must be defined in a user program. Because they differ, this process is described separately for the 7900 and 7905 discs.

For both the 7900 and 7905, when a program tries to access a track by a track number greater than the number of tracks assigned to a given subchannel, the driver sets bit 5 in the status word (end-of-disc) and exits with the transmission log set to the number of tracks assigned to the subchannel. To obtain this information, a program can request an impossible track number once and thereafter stay within the bounds on the subchannel.

If a parity error occurs during disc transfer, a special error message is printed (see Appendix E). For peripheral disc transfers, a parity error causes the transmission log to be returned to the calling program as -1.

7900 EXTRA CONTROLLER TRACK CONFIGURATION

The track map table used for a 7900 disc system must contain the following:

- Number of sectors per logical track
- First track number on subchannels 0 through 7
- Number of tracks on subchannels 0 through 7

The information needed to properly configure a disc is fully described in Section VI. The most necessary information is recapitulated here.

The 7900 Disc Drive has a maximum of 203 tracks per platter. The two platters on each drive are divided as follows:

128 words per sector
48 sectors per track
203 tracks per platter

The RTE 7900 Disc Driver treats a logical track as:

64 words per sector
96 sectors per track

SUBCHANNELS

The moving head driver for an HP 7900 disc system can have four drives chained to a single controller. There may be two platters per drive, and each disc platter is a subchannel accessed through a logical unit number that is referenced back to the equipment table (EQT) entry number of the controller. Thus, the disc system can control a maximum of eight subchannels, numbered 0 through 7.

Subchannels are numbered so that even-numbered subchannels are fixed platters and odd numbered subchannels are removable platters.

SECTORS

READ DATA. The drivers divide each track into 64-word sectors. Whenever more than 64 words are transmitted, the READ request is fastest when begun on an even sector.

WRITE DATA. WRITE requests starting on an odd sector or ending in an even sector require more time; thus, the fastest transfers are WRITE requests that start on an even sector and end in an odd sector. The system always organizes programs and swaps them out in such a way that transfers start on an even and end on an odd sector, thereby minimizing program load and swap times. The WRITE request data can be checked for recoverability by setting bit 10 in the control word (ICNWD). This check on all data written slows the WRITE process.

TRACKS

Each subchannel may contain from 0 to 203 tracks. 203 tracks are the maximum available on the 7900 physical disc. The first track may be any track on the platter. Tracks available to the driver are numbered relative to the first track assigned to the system on each subchannel; thus, if the first available physical track on a subchannel is 10, access by the user to this track must specify logical track number 0.

DEFINING 7900 TRACK MAP TABLE

When an extra controller is used, tracks can only be mapped by defining a table in the user program as follows:

```
ASMB,R,B,L
        NAM   $TB31,0
        END   $TB32
$TB31   DEC   -n
        DEC   FT0,FT1,FT2,FT3,FT4,FT5,FT6,FT7
        DEC   NO0,NO1,NO2,NO3,NO4,NO5,NO6,NO7
        END
```

where *n* is the number of 64-word sectors per track
FT0 through FT7 are the first track numbers for each subchannel 0 through 7
NO0 through NO7 are the number of tracks on subchannels 0 through 7

Example:

Assume a 7900 disc with two subchannels, 0 and 1. Place tracks 0 through 100 on subchannel 0 and tracks 20 through 80 on subchannel 1.

```
ASMB,R,B,L
        NAM   $TB31,0
        ENT   $TB31
$TB31   DEC   -96       96 sectors per track
        DEC   0,20,0,0,0,0,0,0
        DEC   101,61,0,0,0,0,0,0
```

7905 EXTRA CONTROLLER TRACK CONFIGURATION

The table used to map the 7905 contains the following information:

- Number of sectors per track
- Total number of subchannels on drive

And for each subchannel, the following must be specified:

- Cylinder number of track 0
- Number of surfaces per cylinder
- Head number of track 0
- Unit number of disc drive
- Number of tracks on subchannel

To properly configure a track on the 7905, certain information is given here; a full description of track configuration can be found in Section VI.

The HP 7905 Disc Drive provides three surfaces per disc drive. Each surface is divided as follows:

128 words per sector
48 sectors per track
411 tracks per surface

The RTE 7905 Disc Driver treats a logical track as:

64 words per sector
96 sectors per track

SUBCHANNELS

The HP 7905 disc system can control up to eight disc drives connected to one controller. Each disc drive consists of two platters of which one surface is reserved leaving three surfaces to record data. Unlike the 7900, the 7905 subchannels are not directly related, one per platter, to the disc drive and the 7905 is not restricted to eight subchannels.

Each subchannel is a contiguous group of tracks on a single drive. There may be more than one subchannel per drive, but subchannels cannot cross drive boundaries. The exact number of subchannels is specified by the user. There may be as many as 32 subchannels per drive. Subchannels are numbered sequentially from zero; no numbers may be skipped.

SECTORS

The discussion of sectors for the 7900 is also true for the 7905.

TRACKS

Each disc drive has 411 cylinders (or head positions) resulting in a maximum of 1,233 tracks (411 head positions times the 3 disc surfaces). Theoretically, this number of tracks could all be assigned to one subchannel, however, there are program limitations. Peripheral disc subchannels used by the Batch-Spool Monitor must not have more than 1024 tracks, excluding spares, per subchannel. On system or auxiliary discs (logical units 2 or 3), each subchannel is limited to 256 tracks excluding spares.

Head positions (cylinders) are numbered from 0 through 410. There is one head for each surface, numbered 0, 1, 2.

SURFACE ORGANIZATION

Subchannels may be on one, two, or three surfaces, one head per surface. It is best to alternate surfaces when more than one surface is used. This minimizes head movement. For example, if track 0 is at cylinder (head position) 10 on head 0, then track 1 should be at cylinder 10 on head 1 and track 2 at cylinder 11 on head 0. The implications of splitting a subchannel between fixed and removable platters are discussed in Section VI under Disc Planning.

UNIT NUMBER

The unit number is a number associated with each 7905 disc drive. It may be set by the user behind the front panel of the drive, and is always displayed on the front panel. There may be eight units, numbered 0 through 7.

DEFINING THE 7905 TRACK MAP TABLE

When an extra controller is needed, tracks are mapped in a table defined as follows:

```
ASMB,R,B,L
        NAME   $TB32,0
        ENT    $TB32
$TB32   DEC    96      number of 64-word sectors must
                       be 96
        DEC    -n      n is the total number of subchannels
SC0     DEC    x       cylinder number of track 0 for sub-
                       channel 0 (SC0)
        OCT    a       a is defined below
        DEC    t       t is the number of tracks this sub-
                       channel
SC1
          .
          .            repeat for next subchannel
          .
SCn-1
          .
          .            until all subchannels are defined
          .
        END
```

Where:

a is defined as:

bits 15 - 12 = number of surfaces per cylinder
bits 11 - 8 = head number of track 0
bits 3 - 0 = unit number of the disc

Spare tracks can be specified by skipping tracks after each subchannel when constructing the table. A good rule of thumb is to have 11 spare tracks for every 400 tracks; this is the same as 11 spare tracks per surface. To skip tracks, set the cylinder number of track 0 for each subchannel to a number greater than the cylinder number of the last track of the next lower subchannel on that surface.

Example:

Define 10 HP 7905 subchannels using two surfaces of the removable disc cartridge. The number of tracks on each subchannel is 76 plus 4 spare tracks per subchannel. Each subchannel starts at head 0. Only the first three subchannel definitions are fully shown in the following code:

```
ASMB,R,B,L
        NAM    $TB32,0
        ENT    $TB32
$TB32   DEC    96
        DEC    -10     total of 10 subchannels
SC0     DEC    0       first subchannel (subchannel 0)
                       starts at cylinder 0
        OCT    20005   two surfaces, head 0, unit 5
        DEC    76      76 tracks for subchannel 0
SC1     DEC    40      Second subchannel starts at
                       cylinder 40 (4 spare tracks)
        OCT    20005
        DEC    76
SC2     DEC    80      third subchannel starts at cylin-
                       der 80 (4 spare tracks)
        OCT    20005
        DEC    76
SC3     DEC    120
  .       .      .     continue for remaining sub-
  .       .      .     channels through SC9
  .       .      .
SC9     DEC    360
        OCT    20005
        DEC    76
        END
```

MULTIPLE CPU/7905 SYSTEM OPERATION

In a multiple CPU/7905 System environment, the 7905 disc drivers and the controller prevent destructive interference during transfers of data to and from the disc. If a CPU is not to share access to the same physical disc addresses with any other CPU, this is adequate protection.

If a file or set of files is to be shared by more than one CPU, a procedure is needed to prevent the following possible events:

a. CPU A reads a sector to update it.

b. CPU B reads the same sector to update it.

c. CPU A writes its updated sector back to the disc.

d. CPU B writes its updated sector back to the disc, destroying the effect of CPU A access.

To allow software to be written to effect multiple CPU/7905 System operation without destructive interference, the HP 7905 driver (DVR32) services a lock/unlock function call. This call can be issued from one CPU to lock the disc during an I/O operation or set of I/O operations. No other CPU can access the disc until an unlock function call is issued by the original CPU.

DVR32 LOCK/UNLOCK FUNCTION CALL

The I/O Control request (see Section III) is used to hold a Resource Number (RN) and, subsequently, to release the RN. The RN must be allocated and set as a global RN prior to issuing the I/O Control request. For a description of Resource Numbering, see Resource Management in Section III.

The FORTRAN IV calling sequence for an I/O Control request containing a lock/unlock function call is:

```
ICODE=3
ICNWD=control word
IRNUM=resource number
CALL EXEC(ICODE,ICNWD,IRNUM)
```

ICNWD defines a one-word octal value containing control information. For DVR32, control word bits 12-6 contain a function code for the following control states:

Function Code (bits 12-6)	Meaning
15	Lock
00	Unlock

IRNUM is specified only for function code 15. IRNUM contains the RN to be cleared when the lock function call is executed. If a lock is currently in effect from another CPU, the calling program is suspended until the disc is available. If the lock is obtained immediately, the I/O Control request completes immediately. If a lock is already in force by this disc controller, the request completes with the RN cleared.

The lock/unlock function codes are provided to alleviate any CPU contention problem. If a CPU wishes to modify the same disc area as another CPU, the following code sequence could be executed from both units to prevent their interfering with each other:

```
ICODE=12B                              Allocate and
CALL RNRQ(ICODE,IRNUM,ISTAT)           set global RN

CALL EXEC(3,IDLU+1500B,IRNUM)          - Issue lock call,
                                         function code
                                         = 15

CALL RNRQ(5,IRNUM,ISTAT)               - Set/clear the
                                         RN
  .                                      Lock is
  .                                      granted by
  .                                      this point

CALL EXEC(1,IDLU, . . . . )            Next, read the
  .                                      disc and
  .                                      modify data
  .

CALL EXEC(2,IDLU, . . . . )            - Then, write it
                                         back.

CALL EXEC(3,IDLU)                      - Now, issue
  .                                      unlock call,
  .                                      function code
  .                                      = 0
```

To use the lock/unlock function, each CPU operating system must support it.

The sequence described previously for CPU A and CPU B using the lock/unlock function would now be:

a_1. CPU A requests a lock from the driver and it is granted (no other CPU has a lock in force).

a_2. CPU A reads a sector to update it.

b_1. CPU B requests a lock from its driver. Because CPU A has a lock, CPU B must wait.

c_1. CPU A writes its updated sector back to the disc.

c_2. CPU A releases its lock.

b_2. CPU B disc driver gets an interrupt from the disc controller informing it that the lock is now available and completes the lock requested by B at step b_1.

b_3. CPU B reads the same sector to update it.

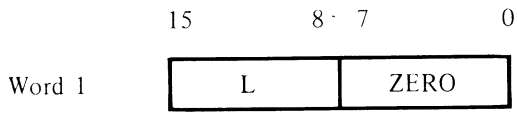
d_1. CPU B writes its updated sector back to the disc. The sector now has both updates.

d_2. CPU B releases its lock.

SOURCE RECORD FORMATS

The source format used for the disc records by the system programs Editor, Assembler, ALGOL, FORTRAN and FORTRAN IV, is given in Table B-1. All records are packed ignoring sector boundaries. Binary records are packed directly onto the disc. After an END record, a zero word is written and the rest of the sector is skipped. If this zero word is the first word of the sector, it is not written. Binary files are always contiguous so a code word is not required.

Table B-1. Source Format

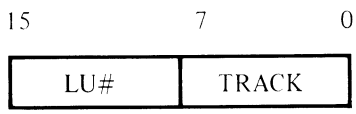


Where L is the record length in words excluding Word 1

If Word 1 = Ø then end of TAPE
If Word 1 = −1 then end of FILE

Odd characters are padded with blanks to make a full word. The last word on any given track in a multi-track file is a code word that points to the next track in the file.

Code Word Format



Where LU# is either 2 (system) or 3 (auxiliary) depending on which platter the track is on.

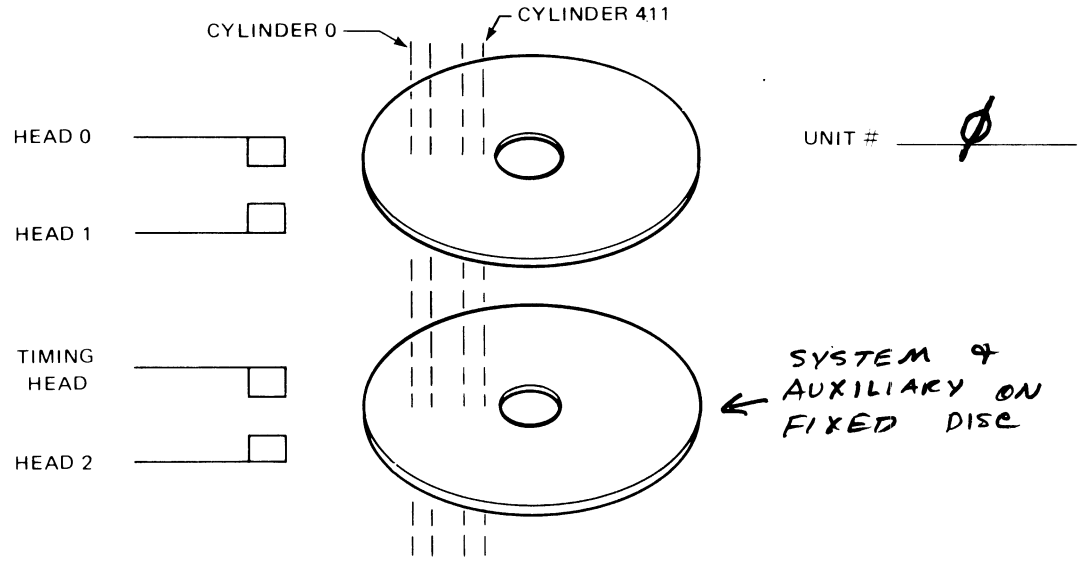
APPENDIX C
SAMPLE RTE-II GENERATION

NOTE

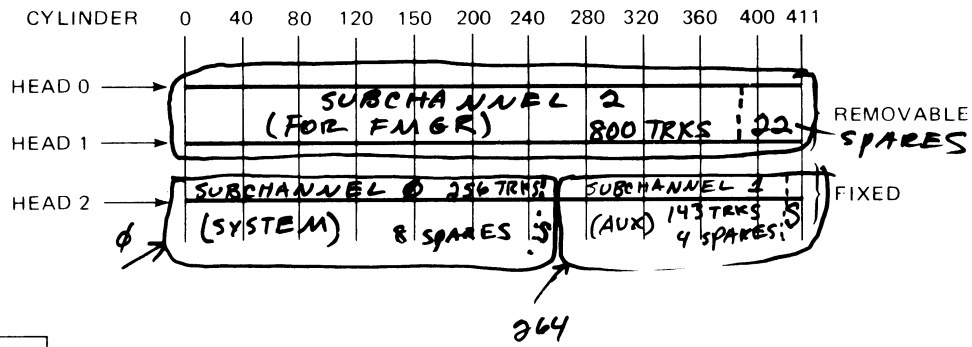
The two completed worksheets included in this Appendix are samples of an HP 7905 Disc Configuration. The generator listing is from an HP 7900 Disc Configuration

Table C-1. Completed Worksheet Giving Suggested 7905 Disc Configuration

STEP 1 **FILL IN UNIT NUMBER:**



STEP 2 **TRACKS SHOWN END-TO-END ON THREE SURFACES—CIRCLE SUBCHANNELS:**

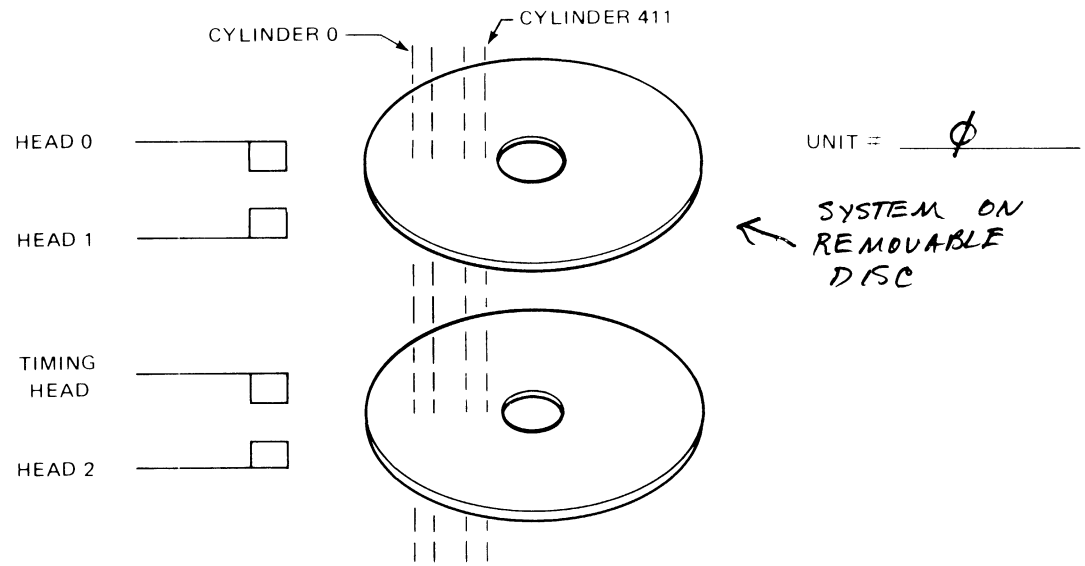


STEP 3 **TRANSLATE STEP 2 TO NUMBERS:**

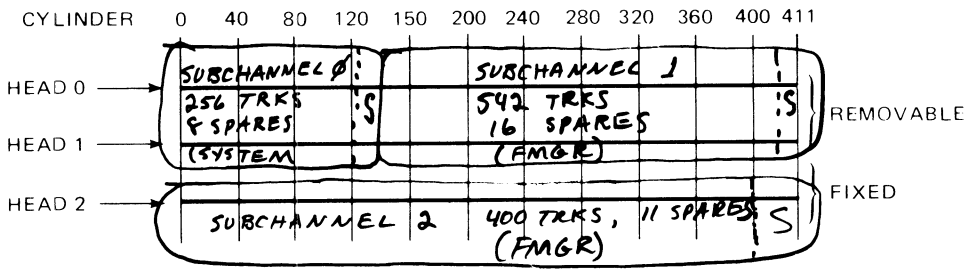
SUBCHANNEL	∅	1	2			
NUMBER OF TRACKS	256	143	800			
STARTING CYLINDER	∅	264	∅			
STARTING HEAD	2	2	∅			
NUMBER OF SURFACES	1	1	2			
NUMBER OF SPARES	8	4	22			
SYSTEM ? (✓)	✓					
AUXILIARY (✓)		✓				
SCRATCH ? (✓)						

Table C-2. Completed Worksheet Giving Suggested 7905 Disc Configuration

STEP 1 FILL IN UNIT NUMBER:



STEP 2 TRACKS SHOWN END-TO-END ON THREE SURFACES–CIRCLE SUBCHANNELS:



STEP 3 TRANSLATE STEP 2 TO NUMBERS:

SUBCHANNEL	Ø	1	2			
NUMBER OF TRACKS	256	542	400			
STARTING CYLINDER	Ø	132	Ø			
STARTING HEAD	Ø	Ø	2			
NUMBER OF SURFACES	2	2	1			
NUMBER OF SPARES	8	16	11			
SYSTEM ? (✓)	✓					
AUXILIARY (✓)						
SCRATCH ? (✓)						

```
MH DISC CHNL?
21

# TRKS, FIRST TRK ON SUBCHNL:
  0?
203,0
  1?
203,0
  2?
203,0
  3?
203,0
  4?
203,0
  5?
203,0
  6?
203,0
  7?
203,0

# 128 WORD SECTORS/TRACK?
48

SYSTEM SUBCHNL?
0

SCRATCH SUBCHNL?
0

AUX DISC (YES OR NO OR # TRKS)?
YES

AUX DISC SUBCHNL?
1

START SCRATCH?
0

TBG CHNL?
13

PRIV. INT. CARD ADDR?
12

FG SWAPPING?
YE

BG SWAPPING?
YE

FG CORE LOCK?
YE

BG CORE LOCK?
YE

SWAP DELAY?
```

```
50

LWA MEM?
77677

PRGM INPT?
MT

LIBR INPT?
PT

PRAM INPT?
TY

INITIALIZE SUBCHNL!
  2?
NO
  3?
NO
  4?
NO
  5?
NO
  6?
NO
  7?
NO

PUNCH BOOT?
YES


PUNCH BOOT?
NO


*EOT


 NO UNDEF EXTS


PARAMETERS

#BSC,14
WHZAT,2,10
MEM,1,32767
SYSON,82,32766
RNRQ,14
LURQ,14
EQLU,14
/E

CHANGE ENTS?

*EAU MACRO!S
.MPY,RP,100200
.DIV,RP,100400
.DLD,RP,104200
.DST,RP,104400
```

```
*HFP MACRO'S
.FAD,RP,105000
.FSB,RP,105020
.FMP,RP,105040
.FDV,RP,105060
IFIX,RP,105100
FLOAT,RP,105120
*FFP MACRO'S
DBLE,RP,105201
SNGL,RP,105202
.XMPY,RP,105203
.XDIV,RP,105204
.DFER,RP,105205
.XADD,RP,105213
.XSUB,RP,105214
.GOTO,RP,105221
..MAP,RP,105222
.ENTR,RP,105223
.ENTP,RP,105224
/E


# OF BLANK ID SEGMENTS?
10

# OF BLANK BG SEG. ID SEGMENTS?
25

FWA BP LINKAGE?
100


SYSTEM

$CSYS(0000)02000 01777  92001-16012 REV.B 741120
BP LINKAGE 00100

DISPA(0000)02000 03165  92001-16012 741120
   *$RENT  02164
   *$BRED  03057
   *$ZZZZ  03115
   *$XEQ   02036
BP LINKAGE 00100

RTIME(0000)03166 03737  92001-16012 741120
   *$TADD  03637
   *$CLCK  03166
   *$TREM  03661
   *$TIME  03370
   *$TIMV  03436
   *$ETTM  03604
   *$TIMR  03532
   *$ONTM  03503
   *$TMRQ  03707
   *$SCLK  03407
   *$BATM  03365
BP LINKAGE 00102

$ASCM(0000)03740 04017  92001-16012 741120
   *$OPER  03770
```

```
   *SERIN  04010
   *SNOPG  04000
   *SILST  03740
   *SNOLG  03751
   *SLGBS  03761
BP LINKAGE 00102

RTIOC(0000)04100 07721  92002-16022  741125
   *SCIC   04100
   *SXSIO  05744
   *SSYMG  07041
   *SIORQ  04267
   *SIOUP  07027
   *SIODN  06776
   *SETEQ  07152
   *SIRT   04216
   *SXCIC  04114
   *SDEVT  06720
   *SGTIO  05264
   *SUPIO  07030
   *SCVEQ  07130
   *SYCIC  04115
   *SBLLO  00102
   *SBLUP  00103
   *SIOCL  07211
   *SLUPR  07326
   *SEQST  07471
   *SCHTO  07557
BP LINKAGE 00113

$ALC (0000)07751 10156  92001-16012 741120
   *$ALC   07751
   *$RTN   10042
BP LINKAGE 00122

EXEC (0000)10162 11716  92001-16012 741120
   *EXEC   10162
   *SERMG  11552
   *SRQST  10164
   *SOTRL  11374
   *SLIBR  10372
   *SLIBX  11053
   *SDREQ  11426
   *SDREL  11512
   *SSDRL  11401
   *SSDSK  11530
   *SERAB  11367
   *SPVCN  10504
   *SREIO  10640
   *SCREL  11270
   *SRSRE  10707
   *SABRE  10774
   *SPWRS  10245
BP LINKAGE 00137

$TRRN(0000)11725 12070  92001-16012 741120
   *STRRN  11725
   *SCGRN  11771
   *SULLU  12013
BP LINKAGE 00145
```

```
$CHED(0000)12134 15174  92002-16012 741125
   *$LIST  12176
   *$MESS  12475
   *$CVT3  14270
   *$CVT1  14334
   *$ABRT  14423
   *$TYPE  14340
   *$MPT1  14465
   *$MPT2  14623
   *$MPT3  14634
   *$MPT4  14724
   *$MPT5  14744
   *$MPT6  14763
   *$PARS  12600
   *$STRT  13425
   *$SCD3  14702
   *$INER  14002
   *$MPT7  15015
   *$ASTM  12531
   *$MPT8  15141
   *$IDNO  15002
   *$WORK  12144
   *$WATR  14613
   *$IDSM  14054
BP LINKAGE 00217

DVP43(0000)15257 15604  92001-16004 REV.B  741028
   *$POWR  15257
   *IP43   15552
   *CP43   15454
BP LINKAGE 00221

DVR00(0000)15622 16701
   *I,00   15622
   *C,00   16171
   *I,01   15622
   *C,01   16171
   *I,02   15622
   *C,02   16171
BP LINKAGE 00221

DVR11(0000)16726 20055
   *C,11   17510
   *I,11   16726
BP LINKAGE 00221

DVR12(0000)20056 20713
   *I,12   20056
   *C,12   20404
BP LINKAGE 00221

DVR23(0000)20714 21557      92202-16001  REV. A
   *I,23   20714
   *C,23   21531
BP LINKAGE 00221

DVR31(0000)21572 22762
   *I,31   22437
   *C,31   22003
```

```
BP LINKAGE 00221

DVR61(0000)23023 24064
   *I,61   23026
   *C,61   23626
BP LINKAGE 00221

DVR62(0000)24114 25020  REV. B
   *C,62   24340
   *I,62   24114
BP LINKAGE 00221

DVR50(0000)25021 25422
   *I,50   25021
   *C,50   25205
   *P,50   25303
BP LINKAGE 00221
  *JP50    25423 25423
   **JP50  25423
BP LINKAGE 00223

$BALB(0000)25424 25423  92002-16006 REV.B 741120
BP LINKAGE 00223

$SPOL(0000)25424 25423  92002-16001 REV.B 741030
BP LINKAGE 00223

DVS43(0000)25461 27743  92002-16003 741205
   *IS43   25461
   *CS43   27115
   *$MPID  25767
   *N.SEQ  27244
BP LINKAGE 00242

SYSLB(0000)27751 27750  92001-16005 REV.B 741120
BP LINKAGE 00242

F4D.C(0000)27751 27750
BP LINKAGE 00242

F2F.B(0000)27751 27750
BP LINKAGE 00242

BP LINKAGE 00242

** OF I/O CLASSES?
16

** OF LU MAPPINGS?
8

** OF RESOURCE NUMBERS?
32

BUFFER LIMITS (LOW, HIGH)?
100,400

* EQUIPMENT TABLE ENTRY

EQT 01?
```

```
21,DVR31,D

EQT 02?
15,DVR00,B,T=32767

EQT 03?
10,DVR50

EQT 04?
17,DVR02,B,T=32767

EQT 05?
16,DVR01,T=32767

EQT 06?
20,DVR12,B,T=32767

EQT 07?
25,DVR00,B,T=32767

EQT 08?
23,DVR23,D,B,T=32767

EQT 09?
26,DVR62,D

EQT 10?
27,DVR61

EQT 11?
14,DVR11,D

EQT 12?
30,DVR00,B,T=32767

EQT 13?
31,DVR00,B,T=32767

EQT 14?
32,DVR00,B,T=32767

EQT 15?
33,DVR00,B,T=32767

EQT 16?
34,DVR00,B,T=32767

EQT 17?
35,DVR00,B,T=32767

EQT 18?
36,DVR00,B,T=32767

EQT 19?
37,DVR00,B,T=32767

EQT 20?
72,DVS43,X=18

EQT 21?
```

```
73,DVS43,X=18

EQT 22?
74,DVS43,X=18

EQT 23?
75,DVS43,X=18

EQT 24?
76,DVS43,X=18

EQT 25?
77,DVS43,X=18

EQT 26?
4,DVP43

EQT 27?
/E


* DEVICE REFERENCE TABLE

 1 = EQT #?
2,0,                SYSTEM CONSOLE

 2 = EQT #?
1,0,                SYSTEM DISC

 3 = EQT #?
1,1,                AUXILIARY DISC

 4 = EQT #?
4,4,                PAPER TAPE PUNCH

 5 = EQT #?
5,0,                PHOTOREADER

 6 = EQT #?
6,0,                LINE PRINTER

 7 = EQT #?
7,0,                BACKGROUND TERMINAL

 8 = EQT #?
8,0,                MAGNETIC TAPE, UNIT 0

 9 = EQT #?
8,1,                MAG TAPE, UNIT 1

10 = EQT #?
8,2,                MAG TAPE, UNIT 2

11 = EQT #?
8,3,                MAG TAPE, UNIT 3

12 = EQT #?
11,0,               CARD READER

13 = EQT #?
```

```
0

14 = EQT #?
1,2,                PERIPHERAL DISC

15 = EQT #?
1,3,                PERIPHERAL DISC

16 = EQT #?
1,4,                PERIPHERAL DISC

17 = EQT #?
1,5,                PERIPHERAL DISC

18 = EQT #?
1,6,                PERIPHERAL DISC

19 = EQT #?
1,7,                PERIPHERAL DISC

20 = EQT #?
12,0,               TERMINAL

21 = EQT #?
13,0,               TERMINAL

22 = EQT #?
14,0,               TERMINAL

23 = EQT #?
15,0,               TERMINAL

24 = EQT #?
16,0,               TERMINAL

25 = EQT #?
17,0,               TERMINAL

26 = EQT #?
18,0,               TERMINAL

27 = EQT #?
19,0,               TERMINAL

28 = EQT #?
0

29 = EQT #?
0

30 = EQT #?
9,0,                H.P. 2313B SUBSYSTEM

31 = EQT #?
10,0,               H.P. 6940 SUBSYSTEM

32 = EQT #?
0

33 = EQT #?
```

```
0

34 = EQT #?
0

35 = EQT #?
0

36 = EQT #?
0

37 = EQT #?
0

38 = EQT #?
0

39 = EQT #?
0

40 = EQT #?
3,0,                RDTS SUBSYSTEM

41 = EQT #?
0

42 = EQT #?
0

43 = EQT #?
0

44 = EQT #?
0

45 = EQT #?
0

46 = EQT #?
0

47 = EQT #?
0

48 = EQT #?
0

49 = EQT #?
0

50 = EQT #?
0

51 = EQT #?
20,0,               SPOOL LU

52 = EQT #?
21,0,               SPOOL LU

53 = EQT #?
```

```
22,0,                SPOOL LU

54 = EQT #?
23,0,                SPOOL LU

55 = EQT #?
24,0,                SPOOL LU

56 = EQT #?
25,0,                SPOOL LU

57 = EQT #?
26,0,                POWER FAIL

58 = EQT #?
/E


* INTERRUPT TABLE
4,ENT,$POWR
10,EQT,3
11,EQT,3
14,EQT,11
15,PRG,PRMPT
16,EQT,5
17,EQT,4
20,EQT,6
21,EQT,1
22,EQT,1
23,EQT,8
24,EQT,8
25,PRG,PRMPT
26,EQT,9
27,EQT,10
30,PRG,PRMPT
31,PRG,PRMPT
32,PRG,PRMPT
33,PRG,PRMPT
34,PRG,PRMPT
35,PRG,PRMPT
36,PRG,PRMPT
37,PRG,PRMPT
72,EQT,20
73,EQT,21
74,EQT,22
75,EQT,23
76,EQT,24
77,EQT,25
/E


BP LINKAGE 00250

 LIB ADDRS 34524
 CHANGE  LIB ADDRS?
36000

LIBRARY

#BSC       )36131 42572
```

```
   **BSC    36131
BP LINKAGE 00431
  #TIME     42627 42633
   **TIME   42627
BP LINKAGE 00432
RNRQ       )42634 43061  92001-16005 741120
   *RNRQ    42634
BP LINKAGE 00433
  $ALRN     43062 43167  92001-16005  741106
   *$ALRN   43062
   *$RNSU   43116
   *$RNEX   43126
   *$LUEX   43142
   *$LUSU   43121
   *$DRAD   43152
BP LINKAGE 00437
LURQ       )43170 43514  92001-16005 741120
   *LURQ    43170
BP LINKAGE 00442
EQLU       )43515 43572  92001-16005 741120
   *EQLU    43515
BP LINKAGE 00443
  #INXT     43573 43612
   **INPT   43573
   **INGT   43603
BP LINKAGE 00443

BP LINKAGE 00443


 FG COMMON 00000
 CHANGE  FG COMMON?
200
RT COM      43613


FG RES ADD 44013
 CHANGE FG RES ADD?
0

FG RESIDENTS

#INRP(0002)44013 44020
   **INRP   44013
BP LINKAGE 00445

EXTND(0010)44021 44206  92002-16004 741030
   *SP,CL   44021
BP LINKAGE 00447
  RMPAR     44207 44231
   *RMPAR   44207
BP LINKAGE 00450
  GETAD     44232 44247
   *GETAD   44232
   *ADRES   44246
BP LINKAGE 00452

SPOUT(0010)44250 45120  92002-16009 741028
BP LINKAGE 00455
  ,DRCT     45121 45127  92001-16005 741120
```

```
   *.DRCT  45121
BP LINKAGE 00456

PRMPT(0010)45130 45240  92001-16003 REV.A  740801
BP LINKAGE 00457

RSPNS(0010)45241 45407  92001-16003 REV.B  741002
BP LINKAGE 00457
  MESSS    45410 45502  92001-16005 741120
   *MESSS  45413
BP LINKAGE 00460

MEM  (2767)45503 45577  VERSION 2 NOV. 1973 -TS
BP LINKAGE 00460

BP LINKAGE 00460


FG DSC ADD 45600
 CHANGE FG DSC ADD?
46000

FG DISC RESIDENTS

#DIAL(0002)46000 46034
   **DIAL  46000
BP LINKAGE 00460

D.RTR(0001)46000 47711  92002-16007  741212
BP LINKAGE 00460
  P.PAS    47712 47740  92002-16006 740801
   *P.PAS  47712
BP LINKAGE 00461
  PRTN     47744 50046  92001-16005  741120
   *PRTM   50037
   *PRTN   47744
BP LINKAGE 00466

SMP  (0030)46105 51141  92002-16002 741205
BP LINKAGE 00465
  OPEN     51165 51352  92002-16006 741205
   *OPEN   51174
BP LINKAGE 00466
  READF    51373 52120  92002-16006  740801
   *READF  51405
   *WRITF  51373
BP LINKAGE 00471
  CLOSE    52136 52244  92002-16006  740801
   *CLOSE  52141
BP LINKAGE 00472
  POST     52245 52273  92002-16006  740801
   *POST   52247
BP LINKAGE 00473
  $OPEN    52274 52502  92002-16006   740801
   *$OPEN  52274
BP LINKAGE 00474
  P.PAS    52503 52531  92002-16006 740801
   *P.PAS  52503
BP LINKAGE 00475
  RW$UB    52532 53003  92002-16006  740801
```

```
   *RWSUB  52532
   *NXSEC  52705
   *SKIP   52625
BP LINKAGE 00477
  RWNDS    53004 53114  92002-16006  740801
   *RWNDS  53006
   *RFLGS  53111
BP LINKAGE 00501
  R/WS     53115 53250  92002-16006  740801
   *R/WS   53115
   *DSXFR  53157
   *D.R    53246
BP LINKAGE 00504
  PRTN     53251 53353  92001-16005  741120
   *PRTM   53344
   *PRTN   53251
BP LINKAGE 00505
  .DRCT    53354 53362  92001-16005 741120
   *.DRCT  53354
BP LINKAGE 00506
  REIO     53363 53465  92001-16005 741120
   *REIO   53367
BP LINKAGE 00507
  RMPAR    53466 53510
   *RMPAR  53466
BP LINKAGE 00510
  GETAD    53511 53526
   *GETAD  53511
   *ADRES  53525
BP LINKAGE 00512

JOB  (0030)46000 47403  92002-16005 741017
BP LINKAGE 00462
  OPEN     47404 47571  92002-16006 741205
   *OPEN   47413
BP LINKAGE 00463
  READF    47610 50335  92002-16006  740801
   *READF  47622
   *WRITF  47610
BP LINKAGE 00465
  CLOSE    50360 50466  92002-16006  740801
   *CLOSE  50363
BP LINKAGE 00466
  POST     50467 50515  92002-16006  740801
   *POST   50471
BP LINKAGE 00467
  $OPEN    50516 50724  92002-16006   740801
   *$OPEN  50516
BP LINKAGE 00470
  P.PAS    50725 50753  92002-16006 740801
   *P.PAS  50725
BP LINKAGE 00471
  RWSUB    50754 51225  92002-16006  740801
   *RWSUB  50754
   *NXSEC  51127
   *SKIP   51047
BP LINKAGE 00473
  RWNDS    51226 51336  92002-16006  740801
   *RWNDS  51230
   *RFLGS  51333
```

```
BP LINKAGE 00475
  R/WS      51337 51472  92002-16006  740801
   *R/WS    51337
   *DSXFR   51401
   *D.R     51470
BP LINKAGE 00500
  .DRCT     51473 51501  92001-16005 741120
   *.DRCT   51473
BP LINKAGE 00501
  REIO      51502 51604  92001-16005 741120
   *REIO    51506
BP LINKAGE 00502
  RMPAR     51605 51627
   *RMPAR   51605
BP LINKAGE 00503
  GETAD     51630 51645
   *GETAD   51630
   *ADRES   51644
BP LINKAGE 00505

AUTOR(0001)46000 46405
   *AUTOR   46000
BP LINKAGE 00460
  TMVAL     46406 46425   92001-16005 741120
   *TMVAL   46410
BP LINKAGE 00461
  FMTIO     46426 47656
   *.RIO.   46672
   *.IIO.   46660
   *.XIO.   46707
   *.XAY.   47045
   *.RAY.   47076
   *.IAY.   47107
   *.DIO.   47215
   *.BIO.   47310
   *.IOI.   47015
   *.IOR.   46754
   *.IAR.   47142
   *.RAR.   47120
   *.DTA.   47376
   *NEWIO   47326
   *OLDIO   47333
BP LINKAGE 00465
  FRMTR     47671 52430
   *.FRMN   50177
   *.LS2F   50171
   *.INPN   50221
   *.DTAN   50213
BP LINKAGE 00620
  .XPAK     52512 52706
   *.XPAK   52523
BP LINKAGE 00621
  .XCOM     52707 52757
   *.XCOM   52707
BP LINKAGE 00622
  .XFER     52760 53023
   *.XFER   52772
BP LINKAGE 00623
  CLRIO     53024 53026
   *CLRIO   53024
```

```
BP LINKAGE 00624
  .FLUN     53027 53047
   *.FLUN   53027
BP LINKAGE 00625
  IAND      53050 53057
   *IAND    53050
BP LINKAGE 00626
  PAUSE     53060 53223
   *.PAUS   53060
   *.STOP   53116
BP LINKAGE 00627
  .ZRLB     53224 53264
   *.ZRLB   53224
BP LINKAGE 00630
  .OPSY     53265 53324
   *.OPSY   53265
BP LINKAGE 00631

WHZAT(0010)46000 47431  RTE-II ONLY J.F.B.741021
BP LINKAGE 00460
  TMVAL     47432 47451   92001-16005 741120
   *TMVAL   47434
BP LINKAGE 00461

SYSON(2766)46000 46034      10 JUL 74 EJW
   *SYSON   46000
BP LINKAGE 00460


BP LINKAGE 00631
 CHANGE BP LINKAGE?
740

 SYS AVMEM 53527
 CHANGE  SYS AVMEM?
53600

BG BOUNDRY 53600
 CHANGE BG BOUNDRY?
0

 BG COMMON 00000
 CHANGE  BG COMMON?
200
BG COM     00200


BG RES ADD 54000
 CHANGE BG RES ADD?
0

BG RESIDENTS

(NONE)


BG DSC ADD 54000
 CHANGE BG DSC ADD?
0
```

```
BG DISC RESIDENTS

#RDTS(0050)54000 61206
   **RDTS  54000
BP LINKAGE 01150
  OPEN     61207 61374  92002-16006 741205
   *OPEN   61216
BP LINKAGE 01151
  READF    61416 62143  92002-16006  740801
   *READF  61430
   *WRITF  61416
BP LINKAGE 01153
  CLOSE    62165 62273  92002-16006  740801
   *CLOSE  62170
BP LINKAGE 01154
  $OPEN    62274 62502  92002-16006   740801
   *$OPEN  62274
BP LINKAGE 01155
  P.PAS    62503 62531  92002-16006 740801
   *P.PAS  62503
BP LINKAGE 01156
  RW$UB    62532 63003  92002-16006  740801
   *RW$UB  62532
   *NX$EC  62705
   *$KIP   62625
BP LINKAGE 01160
  RWND$    63004 63114  92002-16006  740801
   *RWND$  63006
   *RFLG$  63111
BP LINKAGE 01162
  R/W$     63115 63250  92002-16006  740801
   *R/W$   63115
   *D$XFR  63137
   *D.R    63246
BP LINKAGE 01165
  REIO     63251 63353  92001-16005 741120
   *REIO   63255
BP LINKAGE 01166
  RMPAR    63354 63376
   *RMPAR  63354
BP LINKAGE 01167
  GETAD    63377 63414
   *GETAD  63377
   *ADRES  63413
BP LINKAGE 01171

FMGR (0090)54000 54752  92002-16008 REV.B 741027
   *IFLG.  54520
   *CAD.   54644
   *FM.AB  54440
   *D.     54744
   *CUSE.  54742
   *PARS.  54645
   *SEG.R  54470
   *P.SEG  54456
   *INI1.  54005
   *INI2.  54012
   *I.BUF  54220
   *O.BUF  54000
   *N.OPL  54531
```

```
    *P.RAM   54543
    *TTY.    54530
    *NO.RD   54521
    *NOCM.   54643
    *ACTV.   54523
    *J.REC   54524
    *J.NAM   54525
    *G0..    54656
    *JRN.    54522
    *.IDAD   54517
    *TL.P    54747
    *TM.VL   54751
    *L.SEG   54452
    *GT.JB   54034
BP LINKAGE 00740
  FM.CM      54754 56602
    *FM.ER   55740
    *OPEN.   56021
    *CLOS.   56320
    *IER.    56440
    *BRKF.   54754
    *MSS.    55641
    *JER.    56452
    *EC.HO   56363
    *CONV.   56467
    *CAMS.   54755
    *C.BUF   55025
    *.TTY    56557
    *CAM.I   55075
    *CAM.O   55315
    *ECM.    55316
    *BUF.    55317
    *.E.R.   55520
    *.R.E.   55522
    *P.TR    55523
    *TMP.    55524
BP LINKAGE 00754
  OPEN       56700 57065  92002-16006 741205
    *OPEN    56707
BP LINKAGE 00755
  CLOSE      57066 57174  92002-16006  740801
    *CLOSE   57071
BP LINKAGE 00756
  $OPEN      57175 57403  92002-16006   740801
    *$OPEN   57175
BP LINKAGE 00757
  RWND$      57404 57514  92002-16006  740801
    *RWND$   57406
    *RFLG$   57511
BP LINKAGE 00760
  R/W$       57515 57650  92002-16006  740801
    *R/W$    57515
    *D$XFR   57557
    *D.R     57646
BP LINKAGE 00762
  .DRCT      57651 57657  92001-16005 741120
    *.DRCT   57651
BP LINKAGE 00763
  IFBRK      57660 57701  92001-16005 741120
    *IFBRK   57660
```

```
BP LINKAGE 00764
  RMPAR     57702 57724
   *RMPAR   57702
BP LINKAGE 00765
  GETAD     57725 57742
   *GETAD   57725
   *ADRES   57741
BP LINKAGE 00767

FMGR0(0099)57743 57750  92002-16008  740801
BP LINKAGE 00771
  PK..      57751 61366
   *PK..    60102
BP LINKAGE 01000
  CR..      61424 62527
   *CR..    61566
BP LINKAGE 01004
  READF     62646 63373  92002-16006  740801
   *READF   62660
   *WRITF   62646
BP LINKAGE 01010
  RWNDF     63374 63455  92002-16006  740801
   *RWNDF   63403
BP LINKAGE 01011
  NAM..     63456 63552  92002-16006 740801
   *NAM..   63457
BP LINKAGE 01012
  P.PAS     63553 63601  92002-16006 740801
   *P.PAS   63553
BP LINKAGE 01013
  RWSUB     63605 64056  92002-16006  740801
   *RWSUB   63605
   *NXSEC   63760
   *SKIP    63700
BP LINKAGE 01015
  LOCK.     64073 64142
   *LOCK.   64104
BP LINKAGE 01016
  FM.UT     64143 65265
   *D.RIO   64620
   *DR.RD   64703
   *D.SDR   64143
   *PK.DR   64343
   *DS.LU   64543
   *D.LT    64544
   *D.LB    64545
   *DS.DF   64546
   *DS.F1   64547
BP LINKAGE 01025
  CREA.     65266 65337
   *CREA.   65275
BP LINKAGE 01026
  CREAT     65340 65615  92002-16006  741022
   *CREAT   65350
BP LINKAGE 01027
  REIO      65616 65720  92001-16005 741120
   *REIO    65622
BP LINKAGE 01030
  COR.A     65721 65734  92001-16005 741120
   *COR.A   65721
```

```
BP LINKAGE 01031

FMGR1(0099)57743 60035  92002-16008  741122
BP LINKAGE 01005
  .PARS     60036 61057
   *.PARS   60137
BP LINKAGE 01015
  C.TAB     61060 61210  92002-16008  740801
   *C.TAB   61060
BP LINKAGE 01016
  CA..      61211 61403  92002-16008  741119
   *CA..    61214
BP LINKAGE 01017
  REA.C     61404 61466
   *REA.C   61410
BP LINKAGE 01023
  EE..      61467 61530
   *EE..    61476
BP LINKAGE 01025
  TR..      61531 61752
   *TR..    61563
BP LINKAGE 01032
  MR..      61753 62107
   *MR..    61772
BP LINKAGE 01036
  SE..      62123 62307
   *SE..    62144
   *GLOBS   62164
BP LINKAGE 01040
  IF..      62310 62520
   *IF..    62347
BP LINKAGE 01041
  AB..      62521 62702
   *AB..    62554
   *ABX..   62651
   *ABT..   62671
BP LINKAGE 01046
  DP..      62703 62742
   *DP..    62722
BP LINKAGE 01050
  READF     62743 63470  92002-16006  740801
   *READF   62755
   *WRITF   62743
BP LINKAGE 01054
  POSNT     63471 63725  92002-16006  740801
   *POSNT   63501
BP LINKAGE 01055
  P.PAS     63726 63754  92002-16006 740801
   *P.PAS   63726
BP LINKAGE 01056
  RWSUB     63757 64230  92002-16006  740801
   *RWSUB   63757
   *NXSEC   64132
   *SKIP    64052
BP LINKAGE 01065
  WRLG      64232 64247  92002-16006 740801
   *WRLG.   64234
   *EFLG.   64244
BP LINKAGE 01067
  CK.SM     64250 64363
```

```
   *CK.SM  64265
BP LINKAGE 01070
  REIO     64364 64466  92001-16005 741120
   *REIO   64370
BP LINKAGE 01071
  INPRS    64467 64563  92001-16005  741119
   *INPRS  64471
BP LINKAGE 01072
  %WRIT    64564 65264
   *%WRIT  64752
   *%WRIF  64646
   *%WBUF  65051
BP LINKAGE 01074
  .OPSY    65265 65324
   *.OPSY  65265
BP LINKAGE 01075

FMGR2(0099)57743 57754  92002-16008  740801
BP LINKAGE 00771
  IN.IT    57755 60676
   *IN.IT  60051
BP LINKAGE 01005
  IN..     60730 62601
   *IN..   61070
BP LINKAGE 01007
  MC..     62703 63212
   *MC..   62743
BP LINKAGE 01010
  RC..     63213 63400
   *RC..   63245
BP LINKAGE 01013
  PU..     63401 63623
   *PU..   63433
BP LINKAGE 01016
  PURGE    63624 63722  92002-16006  740801
   *PURGE  63631
BP LINKAGE 01017
  NAM..    63727 64023  92002-16006 740801
   *NAM..  63730
BP LINKAGE 01031
  J.PUT    64027 64053  92002-16006  740801
   *J.PUT  64032
BP LINKAGE 01032
  IPUT     64054 64074  92002-16006   740801
   *IPUT   64056
BP LINKAGE 01033
  FID.     64075 64214
   *FID.   64114
BP LINKAGE 01034
  MSC.     64215 64251
   *MSC.   64223
BP LINKAGE 01035
  LOCK.    64252 64321
   *LOCK.  64263
BP LINKAGE 01036
  FM.UT    64322 65444
   *D.RIO  64777
   *DR.RD  65062
   *D.SDR  64322
   *PK.DR  64522
```

```
   *DS.LU  64722
   *D.LT   64723
   *D.LB   64724
   *DS.DF  64725
   *DS.F1  64726
BP LINKAGE 01047
  CNT.     65445 65665
   *CNT.   65512
BP LINKAGE 01050
  FCONT    65666 65757  92002-16006 740801
   *FCONT  65676
BP LINKAGE 01051

FMGR3(0099)57743 57750  92002-16008   740801
BP LINKAGE 00771
  LI..     57751 61410
   *LI..   60315
BP LINKAGE 01002
  DL..     61412 62631
   *DL..   61630
BP LINKAGE 01006
  READF    62736 63463  92002-16006  740801
   *READF  62750
   *WRITF  62736
BP LINKAGE 01012
  LOCF     63464 63651  92002-16006  740801
   *LOCF   63475
BP LINKAGE 01013
  P.PAS    63652 63700  92002-16006 740801
   *P.PAS  63652
BP LINKAGE 01014
  RWSUB    63710 64161  92002-16006  740801
   *RWSUB  63710
   *NXSEC  64063
   *SKIP   64003
BP LINKAGE 01023
  MSC.     64163 64217
   *MSC.   64171
BP LINKAGE 01024
  FM.UT    64220 65342
   *D.RIO  64675
   *DR.RD  64760
   *D.SDR  64220
   *PK.DR  64420
   *DS.LU  64620
   *D.LT   64621
   *D.LB   64622
   *DS.DF  64623
   *DS.F1  64624
BP LINKAGE 01033
  REIO     65343 65445  92001-16005 741120
   *REIO   65347
BP LINKAGE 01034

FMGR4(0099)57743 57757  92002-16008   740801
BP LINKAGE 00771
  ST.DU    57760 61214
   *DU..   60120
   *ST..   60102
BP LINKAGE 01000
```

```
  CO..      61245 61747
   *CO..    61353
BP LINKAGE  01002
  F.UTM     61750 62116
   *LL..    61765
   *LO..    62043
   *SV..    62070
BP LINKAGE  01010
  OPMES     62132 62332
   *PA..    62200
   *TE..    62264
   *AN..    62304
BP LINKAGE  01014
  CREAT     62333 62610  92002-16006  741022
   *CREAT   62343
BP LINKAGE  01015
  READF     62611 63336  92002-16006  740801
   *READF   62623
   *WRITF   62611
BP LINKAGE  01021
  RWNDF     63337 63420  92002-16006  740801
   *RWNDF   63346
BP LINKAGE  01022
  LOCF      63421 63606  92002-16006  740801
   *LOCF    63432
BP LINKAGE  01023
  NAM..     63607 63703  92002-16006 740801
   *NAM..   63610
BP LINKAGE  01024
  P.PAS     63704 63732  92002-16006 740801
   *P.PAS   63704
BP LINKAGE  01025
  RW$UB     63737 64210  92002-16006  740801
   *RW$UB   63737
   *NX$EC   64112
   *$KIP    64032
BP LINKAGE  01034
  FM.UT     64215 65337
   *D.RIO   64672
   *DR.RD   64755
   *D.SDR   64215
   *PK.DR   64415
   *DS.LU   64615
   *D.LT    64616
   *D.LB    64617
   *DS.OF   64620
   *DS.F1   64621
BP LINKAGE  01041
  CREA.     65340 65411
   *CREA.   65347
BP LINKAGE  01042
  CK.SM     65412 65525
   *CK.SM   65427
BP LINKAGE  01043
  REIO      65526 65630  92001-16005 741120
   *REIO    65532
BP LINKAGE  01044

FMGR5(0099)57743 57752  92002-16006   740801
BP LINKAGE  00771
```

```
  ??..      57755 62070  92002-16008  740801
   *??..    57757
BP LINKAGE  01007
  RU..      62071 62531
   *RU..    62207
BP LINKAGE  01012
  RP..      62532 63674
   *RP..    62655
BP LINKAGE  01014
  TL..      63675 63714
   *TL..    63700
BP LINKAGE  01015
  READF     63720 64445  92002-16006  740801
   *READF   63732
   *WRITF   63720
BP LINKAGE  01021
  LOCF      64500 64665  92002-16006  740801
   *LOCF    64511
BP LINKAGE  01022
  P.PAS     64666 64714  92002-16006 740801
   *P.PAS   64666
BP LINKAGE  01023
  RWSUB     64715 65166  92002-16006  740801
   *RWSUB   64715
   *NXSEC   65070
   *SKIP    65010
BP LINKAGE  01025
  J.PUT     65167 65213  92002-16006  740801
   *J.PUT   65172
BP LINKAGE  01026
  IPUT      65214 65234  92002-16006   740801
   *IPUT    65216
BP LINKAGE  01027
  ID.A      65235 65324
   *ID.A    65250
BP LINKAGE  01030
  BUMP.     65325 65363  92002-16006  741025
   *BUMP.   65327
BP LINKAGE  01032
  SET.T     65364 65412  92002-16006 740801
   *SET.T   65366
BP LINKAGE  01033
  TL.       65413 65442  92002-16006  741025
   *TL.     65413
BP LINKAGE  01034
  ST.TM     65443 65476  92002-16006 740801
   *ST.TM   65445
BP LINKAGE  01035
  REIO      65477 65601  92001-16005 741120
   *REIO    65503
BP LINKAGE  01036

FMGR6(0099)57743 57753  92002-16008   740801
BP LINKAGE  00771
  CN..      57754 60014
   *CN..    57763
BP LINKAGE  00775
  JO..      60017 61114
   *JO..    60114
   *SPOL.   60021
```

```
   *JOBFL  60023
BP LINKAGE 01007
  EO..     61115 61740
   *EO..   61217
BP LINKAGE 01020
  OF..     61741 62034
   *OF..   61763
BP LINKAGE 01021
  LG..     62044 62140
   *LG..   62057
BP LINKAGE 01023
  NAMF     62141 62312  92002-16006  740801
   *NAMF   62152
BP LINKAGE 01024
  READF    62313 63040  92002-16006  740801
   *READF  62325
   *WRITF  62313
BP LINKAGE 01030
  POST     63041 63067  92002-16006  740801
   *POST   63043
BP LINKAGE 01031
  NAM..    63070 63164  92002-16006 740801
   *NAM..  63071
BP LINKAGE 01032
  P.PAS    63165 63213  92002-16006 740801
   *P.PAS  63165
BP LINKAGE 01033
  RW$UB    63214 63465  92002-16006  740801
   *RW$UB  63214
   *NX$EC  63367
   *$KIP   63307
BP LINKAGE 01035
  SPOPN    63466 63536  92002-16006  741025
   *SPOPN  63470
BP LINKAGE 01036
  SET.T    63537 63565  92002-16006 740801
   *SET.T  63541
BP LINKAGE 01040
  ST.TM    63566 63621  92002-16006 740801
   *ST.TM  63570
BP LINKAGE 01041
  B.FLG    63622 63670  92002-16006 741118
   *B.FLG  63623
BP LINKAGE 01042
  LULU.    63671 63764  92002-16006 740801
   *LULU.  63673
BP LINKAGE 01043
  RANGE    63766 64011  92002-16006 740801
   *RANGE  63770
BP LINKAGE 01044
  ONOFF    64017 64404  92002-16006 740801
   *ONOFF  64021
   *FIT.   64227
BP LINKAGE 01046
  EX.TM    64405 64573  92002-16006 741008
   *EX.TM  64405
BP LINKAGE 01050
  IPUT     64574 64614  92002-16006   740801
   *IPUT   64576
BP LINKAGE 01051
```

```
  FREE.     64615 64664  92002-16006 740801
   *FREE.   64617
BP LINKAGE  01052
  LU.CL     64665 64726  92002-16006  740801
   *LU.CL   64665
BP LINKAGE  01053
  AVAIL     64727 64771  92002-16006 740801
   *AVAIL   64732
BP LINKAGE  01054
  REIO      64772 65074  92001-16005 741120
   *REIO    64776
BP LINKAGE  01055
  KCVT      65075 65107  92001-16005 741120
   *KCVT    65076
BP LINKAGE  01056
  MESSS     65110 65202  92001-16005 741120
   *MESSS   65113
BP LINKAGE  01057

FMGR7(0099)57743 57752  92002-16008   740801
BP LINKAGE  00771
  CL..      57753 60234
   *CL..    60021
BP LINKAGE  01000
  NX.JB     60242 61107
   *NX.JB   60347
BP LINKAGE  01011
  LU..      61112 62161
   *LU..    61236
BP LINKAGE  01015
  CS..      62213 62405
   *CS..    62260
BP LINKAGE  01016
  READF     62406 63133  92002-16006  740801
   *READF   62420
   *WRITF   62406
BP LINKAGE  01022
  FSTAT     63134 63160  92002-16006  740801
   *FSTAT   63135
BP LINKAGE  01023
  POST      63161 63207  92002-16006  740801
   *POST    63163
BP LINKAGE  01024
  P.PAS     63210 63236  92002-16006 740801
   *P.PAS   63210
BP LINKAGE  01025
  RWSUB     63237 63510  92002-16006  740801
   *RWSUB   63237
   *NXSEC   63412
   *SKIP    63332
BP LINKAGE  01027
  SPOPN     63511 63561  92002-16006  741025
   *SPOPN   63513
BP LINKAGE  01030
  B.FLG     63562 63630  92002-16006 741118
   *B.FLG   63563
BP LINKAGE  01031
  LULU.     63631 63724  92002-16006 740801
   *LULU.   63633
BP LINKAGE  01032
```

```
  RANGE     63725 63750  92002-16006 740801
   *RANGE   63727
BP LINKAGE 01033
  AVAIL     63753 64015  92002-16006 740801
   *AVAIL   63756
BP LINKAGE 01037
  REIO      64024 64126  92001-16005 741120
   *REIO    64030
BP LINKAGE 01040
  KCVT      64127 64141  92001-16005 741120
   *KCVT    64130
BP LINKAGE 01041

FMGR8(0099)57743 57751  92002-16008   740801
BP LINKAGE 00771
  SA..      57752 60724
   *SA..    60066
BP LINKAGE 01000
  SP..      60757 61576
   *SP..    61055
BP LINKAGE 01001
  MS..      61577 62072
   *MS..    61653
BP LINKAGE 01005
  READF     62112 62637  92002-16006  740801
   *READF   62124
   *WRITF   62112
BP LINKAGE 01011
  RWNDF     62640 62721  92002-16006  740801
   *RWNDF   62647
BP LINKAGE 01012
  LOCF      62722 63107  92002-16006  740801
   *LOCF    62733
BP LINKAGE 01013
  P.PAS     63110 63136  92002-16006 740801
   *P.PAS   63110
BP LINKAGE 01014
  RW$UB     63137 63410  92002-16006  740801
   *RW$UB   63137
   *NX$EC   63312
   *$KIP    63232
BP LINKAGE 01016
  IPUT      63411 63431  92002-16006   740801
   *IPUT    63413
BP LINKAGE 01017
  CREA.     63432 63503
   *CREA.   63441
BP LINKAGE 01020
  CREAT     63504 63761  92002-16006  741022
   *CREAT   63514
BP LINKAGE 01021
  NAM..     63763 64057  92002-16006 740801
   *NAM..   63764
BP LINKAGE 01024
  CK.SM     64061 64174
   *CK.SM   64076
BP LINKAGE 01025
  ID.A      64175 64264
   *ID.A    64210
BP LINKAGE 01026
```

```
  WRISS     64265 64323  92002-16006 740801
   *WRIS    64270
   *IWRIS   64310
   *WEOFS   64320
BP LINKAGE 01031
  READ.     64324 64350  92002-16006  740801
   *READ.   64330
BP LINKAGE 01032
  PRTN      64351 64453  92001-16005  741120
   *PRTM    64444
   *PRTN    64351
BP LINKAGE 01033
  REIO      64454 64556  92001-16005 741120
   *REIO    64460
BP LINKAGE 01034
  %WRIS     64557 65046
   *%WRIS   64623
   *%WRIN   64574
   *%WEOF   64714
BP LINKAGE 01037
  SREAD     65047 65604
   *%READ   65047
   *%JFIL   65520
   *%ROSC   65471
BP LINKAGE 01040
  .OPSY     65605 65644
   *.OPSY   65605
BP LINKAGE 01041

GASP (0080)54000 55375
  OPEN      55376 55563  92002-16006 741205
  READF     55564 56311  92002-16006  740801
  CLOSE     56312 56420  92002-16006  740801
  POST      56421 56447  92002-16006  740801
  $OPEN     56450 56656  92002-16006   740801
  P.PAS     56657 56705  92002-16006 740801
  RW$UB     56706 57157  92002-16006  740801
  RWND$     57160 57270  92002-16006  740801
  R/W$      57271 57424  92002-16006  740801
  G1CEX     57425 57526
  ST.LU     57527 57656  92002-16001 741025
  G1ROT     57657 60006  92002-16001  741027
  G1CDJ     60007 60373
  G1CCJ     60374 60735
  G1CD$     60736 62265  92002-16001   741030
  G1C??     62266 63076  92002-16001 741027
  G1STM     63077 63271  92002-16001  740807
  G0QIP     63272 63505  92002-16001  741007
  G1CK$     63506 64275
  G1CIN     64276 65557
  CREAT     65560 66035  92002-16006  741022
  NAM..     66036 66132  92002-16006 740801
  G1CDA     66133 66507
  PURGE     66510 66606  92002-16006  740801
  .DRCT     66607 66615  92001-16005 741120
  REIO      66616 66720  92001-16005 741120
  KCVT      66721 66733  92001-16005 741120
  PARSE     66734 66753  92001-16005 741120
  CNUMD     66754 66773  92001-16005 741120
  RMPAR     66774 67016
```

```
  GETAD     67017 67034

EDITR(0050)54000 60424   92002-16010 REV.B   741210
  CREAT     60425 60702   92002-16006   741022
  OPEN      60703 61070   92002-16006 741205
  READF     61071 61616   92002-16006   740801
  CLOSE     61617 61725   92002-16006   740801
  NAM..     61726 62022   92002-16006 740801
  $OPEN     62023 62231   92002-16006    740801
  P.PAS     62232 62260   92002-16006 740801
  RW$UB     62261 62532   92002-16006   740801
  RWND$     62533 62643   92002-16006   740801
  R/W$      62644 62777   92002-16006   740801
  PRTN      63000 63102   92001-16005   741120
  REIO      63103 63205   92001-16005 741120
  RMPAR     63206 63230
  GETAD     63231 63246

LOADR(0090)54000 64704     92001-16002 REV.B 741210
  PRTN      64705 65007   92001-16005   741120

ASMB (0099)54000 60644   92001-16006 REV.B   741120

ASMBD(0099)60645 61452   92001-16007 741120

ASMB1(0099)60645 62264   92001-16008   741120

ASMB2(0099)60645 62442   92001-16009 741120

ASMB3(0099)60645 61551   92001-16010 741120

ASMB4(0099)60645 62160   92001-16011 741120

FTN4 (0099)54000 70050

F4.0 (0099)70051 75721

F4.1 (0099)70051 73612

F4.2 (0099)70051 75341

F4.3 (0099)70051 75347

ALGOL(0099)54000 66367
  $READ     66370 67125
  %WRIT     67126 67626
  .OPSY     67627 67666

ALGL1(0099)67667 70332

BP LINKAGE 01430

SYSTEM STORED ON DISC
SYS SIZE: 26 TRK$, 028 SEC$(10)
```

APPENDIX D
SUMMARY OF EXEC CALLS

ASSEMBLY LANGUAGE FORMAT

```
    EXT  EXEC        Used to link program
                     to RTE-II
     .
     .
     .
    JSB  EXEC        Transfer control to
                     RTE-II

    DEF  *+n+1       Defines point of
                     return from RTE-II, 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 in-
                     direct
    DEF  pn

    return point     Continue execution
                     of program

p1  -
     .
     .               Actual parameter
     .               values
pn  -
```

For each EXEC call, this appendix includes only the parameters (*p1* through *pn* in the format above) of the Assembler Language calling sequence.

FORTRAN/FORTRAN IV FORMAT

```
CALL EXEC(7)
  -or-               Equivalent
ICODE = 7            calling sequences
CALL EXEC(ICODE)

CALL EXEC(ICODE,p2...pn)
```

Where:

p2 through *pn* are either integer values or integer variables defined elsewhere in the program.

Note that some EXEC call functions are handled automatically by the FORTRAN compilers or special subroutines. Refer to FORTRAN, Part 3, Section IV and the specific EXEC calls.

READ/WRITE

Purpose:

Transfers input or output.

Assembly:

ICODE DEC	1 = READ, 2 = WRITE 17 = Class READ 18 = Class WRITE 20 = Class WRITE/READ
ICNWD OCT	Control Word, see Section III.
IBUFR BSS	Buffer of *n* words
IBUFL DEC	Same *n*; words (+), characters (−)
IPRM1 DEC *p*	optional parameter. Used for disc track in disc call.
IPRM2 DEC *q*	optional parameter. Used for disc sector in disc call.
ICLAS OCT	Class Word, see Section III.

FORTRAN:

```
REG=EXEC(ICODE,ICNWD,IBFR,IBFL,IP1,IP2,ICLS)
```

I/O CONTROL

Purpose:

Carry out control operations

Assembly:

ICODE DEC 3 or 19	3 = Control 19 = Class Control
ICNWD OCT	Control Word, see Section III.
IPRAM DEC *n*	(Optional parameter required by some CONWDs)
ICLAS OCT	Class Word, see Section III.

FORTRAN:

```
REG=EXEC(ICODE,ICNWD,IPRAM,ICLAS)
```

CLASS I/O - GET

Purpose:

Request device status.

Assembly:

ICODE DEC 21	
ICLAS NOP	Class Word, see Section III
IBUFR BSS	Buffer of *n* words
IBUFL DEC	Same *n*; words (+), characters (−)
IRTN1 NOP	Return for IPRM1
IRTN2 NOP	Return for IPRM2
IRTN3 NOP	Return for ICLAS

FORTRAN:

```
CALL EXEC(ICODE,ICLAS,IBUFR,IBUFL,IR1,
IR2,IR3)
```

I/O STATUS

Purpose:

Request device status.

Assembly:

ICODE DEC 13	
ICNWD DEC *n*	Logical unit number
IEQT5 NOP	Word 5 of EQT entry returned here
IEQT4 NOP	Optional parameter for word 4 of EQT

FORTRAN:

```
CALL EXEC(ICODE,ICNWD,IEQT5,IEQT4)
```

DISC TRACK ALLOCATION

Purpose:

Request allocation of contiguous tracks.

Assembly:

ICODE DEC 4 or 15	4 = Allocate track to program, or 15 = allocate track globally.
ITRAK DEC *n*	Number of contiguous tracks desired. If bit 15 = 1, do not suspend until available.
ISTRK NOP	Starting track returned here, or −1, not available.
IDISC NOP	Disc logical unit returned here.
ISECT NOP	Number of 64 word sectors returned here.

FORTRAN:

```
CALL EXEC (ICODE,ITRAK,ISTRK,IDISC,ISECT)
```

DISC TRACK RELEASE

Purpose:

Release some disc tracks assigned to the program.

Assembly:

ICODE DEC 5 or 16	5 = Release program's tracks, or 16 = release global tracks.
ITRAK DEC *n*	If = −1, release all program tracks. If = *n*, the number of contiguous tracks starting at ISTRK.
ISTRK NOP	Starting track number.
IDISC NOP	Logical unit.

FORTRAN:

CALL EXEC(ICODE,ITRAK,ISTRK,IDISC)

PROGRAM COMPLETION

Purpose:

Signal end of program.

Assembly:

ICODE DEC 6	
INAME ASC 3,*name*	Name of program to be terminated (0 if this one).
INUMB DEC	0 = Normal completion −1 = Serial reusability 1 = Make dormant but save suspension point 2 = Terminate on next schedule; save tracks. 3 = Terminate immediately and release tracks.
IPRM1 : IPRM5	Up to 5 optional parameters

FORTRAN:

REG = EXEC (ICODE,INAME,INUMB,IPRM1 . . IPRM5)

CALL RMPAR (IPRM1 . . IPRM5) pram pick-up

PROGRAM SUSPEND

Purpose:

Suspend calling program.

Assembly:

ICODE DEC 7

FORTRAN:

PAUSE library subroutine generates this call.

PROGRAM SCHEDULE

Purpose:

To schedule another program.

Assembly:

ICODE DEC 9 or 10 — 9 = Immediate, wait; 10 = Immediate, no wait; 23 = Queue, wait; 24 = Queue, no wait

INAME ASC 3,*xxxxx* — *xxxxx* is the program name

IPRM1
.
. Up to 5 optional parameters
IPRM5

FORTRAN:

REG = EXEC(ICODE,INAME,IPRM1 . . . IPRM5)

PROGRAM SEGMENT LOAD

Purpose:

Load segment of calling program.

Assembly:

ICODE DEC 8

INAME ASC 3,*xxxxx* — *xxxxx* is segment name

IPRM1
.
. Up to 5 optional parameters
IPRN5

FORTRAN:

REG = EXEC (ICODE,INAME,IPRM1 . . IPRM5)

TIME REQUEST

Purpose:

Request the 24-hour time and day.

Assembly:

ICODE DEC 11

ITIME BSS 5 — Time values: tens of milliseconds, seconds, minutes, hours, day, returned in that order.

IYEAR BSS1 — Year (optional)

FORTRAN:

CALL EXEC(ICODE,ITIME,IYEAR)

TIMED EXECUTION (Initial Offset)

Purpose:

Schedule a program to start after a delay.

Assembly:

```
ICODE   DEC 12
IPROG   DEC 0          Schedule calling program,
        ASC 3,name     or Schedule name
IRESL   DEC x          Resolution code
MTPLE   DEC y          Execution multiple
IOFST   DEC -z         z (units set by x) equals
                       the initial offset (neg-
                       ative value)
```

FORTRAN:

```
CALL EXEC(ICODE,IPROG,IRESL,MTPLE,IOPST)
```

TIMED EXECUTION (Absolute Start)

Purpose:

Schedule a program to start at a particular time.

Assembly:

```
ICODE   DEC 12
IPRG    DEC 0          Schedule calling program, or
        ASC 3,name     Schedule name
IRL     DEC x          Resolution code
MT      DEC y          Execution multiple
IHRS    DEC a
MINS    DEC b          Defines absolute start-time
ISECS   DEC c
MSECS   DEC d
```

FORTRAN:

```
CALL EXEC(ICODE,IPRG,IRL,MT,IH,MI,IS,MS)
```

PROGRAM SWAPPING CONTROL

Purpose:

Allows program to lock itself into core.

Assembly:

```
ICODE  DEC 22
IOPTN  DEC      0 = swap OK
                1 = swap not OK
                2 = swap program only
                3 = swap all disc resident area.
```

FORTRAN:

```
CALL EXEC (ICODE,IOPTN)
```

RESOURCE MANAGEMENT

Purpose:

Allows cooperating programs to manage resources.

Assembly:

```
        JSB     RNRQ
        :
ICODE   OCT     Control word, see Section III.
IRN     BSS1    Resource number
ISTAT   BSS1    Status of resource
```

FORTRAN:

```
CALL RNRQ(ICODE,IRN,ISTAT)
```

LOGICAL UNIT LOCK

Purpose:

Locks an I/O device.

Assembly:

```
        JSB     LURQ
        :
IOPTN   OCT     0x000=unlock specified lu
                1x0000=unlock all lu's
                0x0001=lock with wait
                1x0001=lock without wait
                (x is no abort bit)
LUARY   DEC     Array of lu's to be locked/unlocked.
NOLU    DEC     Number of lu's to be locked/unlocked.
```

FORTRAN

```
CALL LURQ(IOPTN,LUARY,NOLU)
```

APPENDIX E
SUMMARY OF ERROR MESSAGES

OPERATOR REQUEST ERROR MESSAGES

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

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

EXEC CALL ERROR MESSAGES

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

When RTE-II aborts a program, it prints the following message:

name ABORTED

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

MP *name address*

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

RQ *name address*

An RQ00 error means that the address of a returned parameter is below the memory protect fence.

The following errors have the same format as "MP" and "RQ" errors.

Error	Meaning
TI	Batch program exceeds allowed time
RE	Re-entrant subroutine attempted recursion (call itself)

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 DISC ALLOCATION CALLS

DR01 = Insufficient number of parameters,

DR02 = Number of tracks is $\leqslant$ zero; illegal logical unit; or number of tracks to release is zero or negative.

DR03 = Attempt to release track assigned to another program

ERROR CODES FOR SCHEDULE CALLS

SC00 = Batch program attempted to suspend (EXEC(7))

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-II ignores the interrupt and returns to the point of interruption.

SC04 = *name* is not a subordinate (or "son") of the program issuing the completion call.

SC05 = Program given is not defined.

SC06 = No resolution code in EXECUTION TIME EXEC call.

SC07 = Prohibited core lock attempted.

ERROR CODES FOR I/O CALLS

IO00 = Illegal class number

IO01 = Not enough parameters

IO02 = Illegal logical unit

IO03 = Not used

IO04 = Illegal user buffer

IO05 = Illegal disc track or sector

IO06 = Reference to a protected track; or using load-and-go before assigning load-and-go tracks (see LG, Section II)

IO07 = Driver has rejected call

IO08 = Disc transfer longer than track

IO09 = Overflow of load-and-go area

ERROR CODES FOR PROGRAM MANAGEMENT CALLS

RN00 = No option bits set in all

RN01 = Resource number not defined

RN02 = Resource number not defined

RN03 = Unauthorized attempt to clear a LOCAL resource number

ERROR CODES FOR LOGICAL UNIT LOCK CALLS

LU01 = Program has one or more logical units locked and is trying to LOCK another with WAIT

LU02 = Illegal logical unit reference (greater than maximum number)

LU03 = Not enough parameters furnished in the call

INPUT/OUTPUT ERROR MESSAGES

ILLEGAL INTERRUPTS

When an illegal interrupt occurs, RTE-II prints this message:

ILL INT *xx*

Where *xx* is the octal channel number.

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

EQUIPMENT ERROR MESSAGES

Message	Meaning
I/O ERR ET EQT # eqt	End-of-tape condition on device # eqt. Correct the condition and set device UP.
I/O ERR TO EQT #eqt	Device #eqt has timed-out. Examine device. Correct problem and set device UP.
I/O ERR NR EQT #eqt	Device #eqt is not ready. Make ready and set device UP.

I/O ERROR PE EQT #eqt — Parity error in data transmission from device #eqt. Examine device.

TR *nnnn* EQT *eqt*, U*pp* S (or U) — Irrecoverable disc transfer parity error. If the transfer is to a system or auxiliary disc the following applies.

Where:
nnn = Track number
eqt = EQT number
pp = Unit or subchannel number

a. If user request (U), then program is abnormally terminated and track is made unavailable for further operations. If the user request was an on-line modification with the RTE-II loader, the parity error could be the result of failing to turn off the hardware disc protect switch. The loader should be executed again with the protect switch off.

b. If system request (S), the program transfer terminates.

If user request to peripheral disc, a transmission log of -1 is returned to the calling system.

RTE EDITOR ERRORS

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

/EDIT: *error message: illegal edit command*

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

Error Message	Meaning
MEM OVERFLOW	The Edit File; overflows available memory; RTE-II Editor prints the command causing the overflow. Edit terminates.
CS ERR	Illegal edit command, which is printed.
PARAM ERR	Edit command "r" or "c" is illegal: non-numeric, = 0, > 72, $r_2 \leq r_1$, $c_2 \leq c_1$, command printed.
SEQ ERR	"r" parameter ≤ a previous "r", or "r" greater than range of Symbolic File; command printed.
/I ERR	No insert source statements after /I; command printed.
/R ERR	No insert source statements after /R; command printed.
/C OVG	Character overflow in edit statement (i.e.,>72 character)
DISK OVF	No disc space for file; edit terminates.
FILE UN	Undefined symbolic File, or LUN (Edit File) = 2. Edit terminates.

RTE INTERACTIVE EDITOR ERRORS

Error Message	Meaning
??	Error in command just entered.
EOF	A command has caused an attempt to read beyond the current end of the source file. A/R*text* or */text* command causes *text* to become the pending line and the EOF then follows the pending line. A */I**text*** command inserts *text* before the EOF, but the file remains with the EOF pending.
CORRUPT FILE	Input record length from the system LS area is greater than 150 characters.
FILE MANAGER ERROR - *nn*	An error detected by the file manager routines. See the Batch Spool Monitor manual for details.

FORTRAN COMPILER ERRORS

More than one source tape can be compiled into one FORTRAN program by leaving off the $END statement on all but the last source tape. When the end of each source tape is encountered (end-of-tape or EOT condition), RTE Driver DVR00 can interpret it in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how DVR00 was configured during generation. In any case, an EOT does not suspend the FORTRAN Compiler. Therefore, it is recommended that when compiling multiple tapes, DVR00 be configured to set the tape reader down on EOT. For more information refer to the DVR00 Manual (HP Part No. 29029-95001).

If an EOT causes the tape reader to be set down, the RTE-II system will output a message to the operator:

I/O ERR ET EQT *#eqt*

The operator must place the next source tape into the tape reader and set the tape reader up with the UP operator command.

UP,*eqt*

If an EOT does not cause the tape reader to be set down, the RTE-II system does not output any message and the compiler is not suspended.

At the end of compilation (when the compiler detects the $END statement), the following message is printed.

$END,FTN

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

IO06

IO09

IO06 means that the load-and-go tracks were not defined by an LG operator request, and IO09 means that the load-and-go tracks overflowed. The operator must define more load-and-go tracks with LG and start compilation over again.

The compiler terminates abnormally if:

a. No source file is declared by LS, although logical unit 2 is given for input. Compiler error E-0019 (FTN2), or ERROR 05 (FTN4) is printed on the list device.

b. The symbol table overflows. Compiler error E-0014 (FTN2), or ERROR 03 (FTN4) is printed on the list device. $END,FTN does not appear after the error message using FTN2, but does appear when using FTN4.

ALGOL ERRORS

More than one source tape can be compiled into one ALGOL program by leaving off the END$ statement on all but the last source tape. When the end of each source tape is encountered (end-of-tape or EOT condition), RTE Driver DVR00 can interpret it in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how DVR00 was configured during generation. In any case, an EOT does not suspend the ALGOL Compiler. Therefore, it is recommended that when compiling with multiple tapes, DVR00 be configured to set the tape reader down on EOT. For more information refer to the DVR00 manual (HP Part No. 29029-95001).

If an EOT causes the tape reader to be set down, the RTE-II system will output a message to the operator:

I/O ERR ET EQT *#eqt*

The operator must place the next source tape into the tape reader and set the tape reader up with the UP operator command.

UP,*eqt*

If an EOT does not cause the tape reader to set down, the RTE-II system does not output any message and the compiler is not suspended.

At the end of compilation (when the compiler detects the END$ statement), the following message is printed.

$END ALGOL

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

NO SOURCE

is printed on the system teleprinter and compilation ceases.

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

name: PAUSE*xxxx*

Where:

name = program name

xxxx = a number which has no significance.

Execution is then suspended. To restart the program type:

GO,*name*[,*p1*,*p1*,*p3*,*p4*,*p5*]

See the GO operator command in Section II for a definition of the parameters.

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

IO06

IO09

IO06 means that the load-and-go tracks were not defined by an LG operator request, and IO09 means that the load-and-go tracks overflowed. The operator must define more load-and-go tracks with LG and start compilation over again.

ASSEMBLER ERRORS

When a paper tape is being input through the tape reader, RTE-II Driver DVR00 can interpret an end-of-tape (EOT) in two ways. An EOT can set the tape reader down (make it inactive), or not set it down. The action depends on how DVR00 was configured during generation. In any case, an EOT does not suspend the Assembler. Therefore, it is recommended that when assembling multiple tapes, DVR00 be configured to set the tape reader down on EOT. For more information refer to the DVR00 manual (HP Part No. 29029-95001).

If an EOT causes the tape reader to be set down, the RTE-II system will output a message to the operator:

I/O ERR ET EQT *#eqt*

The operator must up the tape reader with the UP operator command.

UP, *eqt*

If an EOT does not cause the tape reader to be set down, the RTE-II system does not output any message and the assembler is not suspended.

At the end of assembly, the following message is printed:

$END ASMB

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

$END ASMB PASS

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

GO,ASMB

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

$END ASMB CS

The current assembly aborts.

If an end-of file condition occurs before an END statement is found (LS File only), the teleprinter signals:

$END ASMB XEND

The current assembly aborts.

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

$END ASMB NPRG

The current assembly aborts.

RTE-II generates two messages when ASMB attempts to write on the load-and-go tracks (RTE-II aborts ASMB).

IO06

IO09

IO06 means that the load-and-go tracks were not defined by an LG operator request, and IO09 means that the load-and-go tracks have overflowed. The operator must define more load-and-go tracks with LG and start

IO06 means that the load-and-go tracks were not defined by an LG operator request, and IO09 means that the load-and-go tracks have overflowed. The operator must define more load-and-go tracks with LG and start compilation over again.

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

#tape numb

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

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

PG *page numb*

page numb is the page number (in the listing) of the previous error diagnostic.

PG 000 is associated with the first error in the program.

These messages occur on a seperate line, above each error diagnostic in the listing.

RELOCATING LOADER ERRORS

Messages are printed in this format:

/LOADR: *message*

"L" ERROR MESSAGES

L01 – checksum error

L02 – illegal record

These errors are recoverable. The offending record can be reread by repositioning the tape and typing:

GO,LOADR

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

L03 – Memory overflow

L04 – Base page linkage area overflow

L05 – Symbol table area overflow

L06 – Common block error

a. Exceeding allocation in a replacement or addition.

b. In a normal background load, first program did not declare largest common block.

L07 – Duplicate entry points

L08 – No transfer address (main program) in the program unit. Another program may be entered with a GO operator request. (This also occurs when load-and-go is specified, but no program exists in the load-and-go area.)

L09 – Record out of sequence

L10 – Operator request parameter error. GO requests may be retyped; RU requests may not.

L11 – Operator attempted to replace or purge a core-resident program.

L12 – LG area used without resetting (*input option* = 2 in "GO"). *Input option* was not input as 99 previously.

L13 – LG area has been illegally reset-overwritten. Program addition on LG area not allowed if it has already been specified for program input. Or LG area was once used for force loading with *input option* = 99 and it is again being used with *input option* = 99.

L14 – ASMB produced illegal relocatable. A DBL record refers to an external which has not been defined (the original can not be found in the symbol table).

L15 – Forward reference to a type 3 or type 4 ENT or to an EXT with offset which has not yet been defined, or a forward indirect external reference.

ADDITIONAL MESSAGES

NO BLANK ID SEGMENT

This message is printed when an available (i.e., blank) ID Segment is not found. The loader calls for program suspension. The operator may then delete a program from the system (OF operator request) or may terminate the loader.

WAITING FOR DISC SPACE

This message is printed when a track allocation cannot be made. The loader repeats the disc request and is suspended until space becomes available.

UNDEFINED EXTS

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

LOAD

This message is printed and the loader is suspended whenever an End-of-Tape condition is detected from the input unit.

DUPLICATE PROG NAME–*name*

This message is printed when a program *name* is already defined in the system for a normal load or a program addition. The loader changes the name of the current program by replacing the first two characters with periods (e.g., JIMB1 becomes . . MB1). The second duplicate program name aborts the loader.

SET PRGM INACTIVE

This message is printed when an end-of-tape condition is detected from the input device being used for library input and the loader needs the library to be scanned again.

At the end of a normal load, or after loading the last segment, the loader prints the following message and terminates itself.

/LOADR: *name* READY
/LOADR: $END

APPENDIX F
SUMMARY OF OPERATOR REQUESTS

AB[,0 / ,1]	Abort current BATCH program. 0 = save tracks 1 = return tracks
BL,*[lower limit, upper limit]*	Sets buffer limits
BR,*name*	Sets a break flag in *name*
DN,*eqt*	Set EQT device *eqt* down.
EQ,*eqt*	Print status of EQT entry *eqt.*
EQ,*eqt*[,*UN* / ,*BU*]	*UN* = delete buffering *BU* = specify buffering
GO,*name [,p1[,,, . . . [,p5]]]]*	Reschedule program *name*
IT,*name[,res,mpt[,hr,min[,sec[,ms]]]]*	Schedule program to execute at specified times.
LG,*numb*	Allocate *numb* L & G tracks or release L & G tracks (*numb* = 0).
LS,*disc lu, trk numb*	Declare disc number *disc lu,* track number *trk numb* as source file.
LU, *lu*	Print status of logical unit *lu.*
LU, *lu*, *eqt* / *0* *[subch numb]*	Assign EQT number *eqt* (subchannel *subch numb*) to LU number *lu.*
OF,*name*[,0 / ,1 / ,8]	Terminate *name* 0 = save tracks, 1 = return tracks, 8 = purge *name*
ON,*name [,NOW] [,pl [,... [,p5]]]]]*	Schedules *name. NOW* means ignore time values.
PR,*name,numb*	Set priority of *name = numb*
RU,*name [,p1[, . . . [,p5]]]]*	Schedule program *name*
RT,*name*	Release *name's* tracks
SS,*name*	Suspend *name*
ST,*name*	Print status of *name*
ST[,0 / ,1 / ,2]	0 = print name of current program. 1 = print name of current FG disc resident program. 2 = print name of current BG disc resident program.

SW	Print status of swapping word.
SW ,0 SW ,1 SW ,2 SW ,3	Change status of swapping word. 0 = no swapping 1 = foreground swapping 2 = background swapping 3 = both foreground/background swapping.
TI	Print current real-time
TM,*year,day[,hr,min,sec]*	Specify year, day and 24 hour time.
TO,*eqt*	Print time-out value of EQT number *eqt.*
TO,*eqt, numb*	Assign time-out value *numb* to EQT number *eqt*
UP,*eqt*	Set EQT number

APPENDIX G
HP CHARACTER SET

b_7 b_6 b_5 / BITS b_4	b_3	b_2	b_1	COLUMN → / ROW ↓	0 (0 0 0)	1 (0 0 1)	2 (0 1 0)	3 (0 1 1)	4 (1 0 0)	5 (1 0 1)	6 (1 1 0)	7 (1 1 1)
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	↑	n	~
1	1	1	1	15	SI	US	/	?	O	(←)	o	DEL

Figure G-1. ASCII Characters and Binary Codes

NOTES:

Standard 7-bit set code positional order and notation are shown below with b_7 the high-order and b_1 the low-order, bit position.

Example: The code for "R" is: $b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1$ = $1\ 0\ 1\ 0\ 0\ 1\ 0$

On some devices, the underscore (—) prints as a left arrow (←).

RTE-II SPECIAL CHARACTERS:

Mnemonic	Value	Use
SOM (Control A)	1	Backspace
S1 (2600 Backspace) (Control Y)	31	Backspace
FE_0 (Control H)	10	Backspace
EOT (Control D)	4	Simulate End Tape

NUL	Null	DC1	Device Control 1
SOH	Start of Heading	DC2	Device Control 2
STX	Start of Text	DC3	Device Control 3
ETX	End of Text	DC4	Device Control 4
EOT	End of Transmission	NAK	Negative Acknowledgement
ENQ	Enquiry	SYN	Synchronous Idle
ACK	Positive Acknowledgement	ETB	End of Transmission Block
BEL	Bell (Audible signal)	CAN	Cancel
BS	Backspace	EM	End of Medium
HT	Horizontal Tabulation	SUB	Substitute
LF	Line Feed	ESC	Escape
VT	Vertical Tabulation	FS	File Separator
FF	Form Feed	GS	Group Separator
CR	Carriage Return	RS	Record Separator
SO	Shift Out	US	Unit Separator
SI	Shift In	SP	Space
DLE	Data Link Escape	DEL	Delete

Table G-1. Legend for Figure G-1.

Table G-2. ASCII/Octal Table

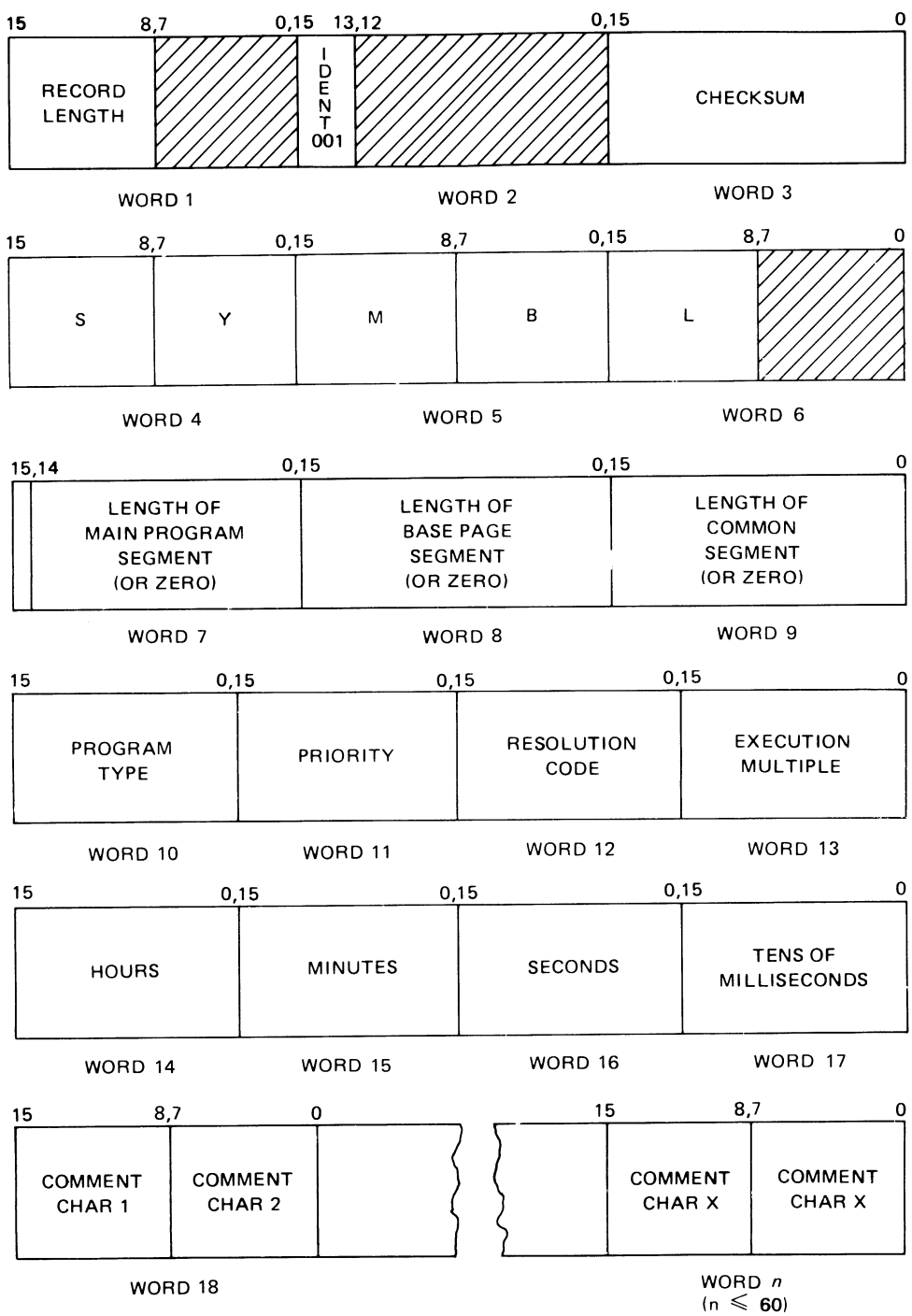
ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
space	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
–	026400	000055

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
.	027000	000056
/	027400	000057
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
↑	057000	000136
_ (←)	057400	000137
ACK	076000	000174
①	076400	000175
ESC	077000	000176
DEL	077400	000177
NULL	000000	000000
SUM	000400	000001
EOA	001000	000002
EOM	001400	000003
EOT	002000	000004
WRU	002400	000005
RU	003000	000006
BELL	003400	000007
FE_0	004000	000010
HT/SK	004400	000011
LF	005000	000012
V_{TAB}	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DC_0	010000	000020
DC_1	010400	000021
DC_2	011000	000022
DC_3	011400	000023
DC_4	012000	000024
ERR	012400	000025
SYNC	013000	000026
LEM	013400	000027
S_0	014000	000030
S_1	014400	000031
S_2	015000	000032
S_3	015400	000033
S_4	016000	000034
S_5	016400	000035
S_6	017000	000036
S_7	017400	000037

APPENDIX H
PAPER TAPE FORMATS

NAM RECORD

CONTENT



EXPLANATION

RECORD LENGTH = 9-60 WORDS

IDENT = 001

CHECKSUM: ARITHMETIC TOTAL OF ALL WORDS IN RECORD EXCLUDING WORDS 1 AND 3.

SYMBL: FIVE CHARACTER NAME OF PROGRAM

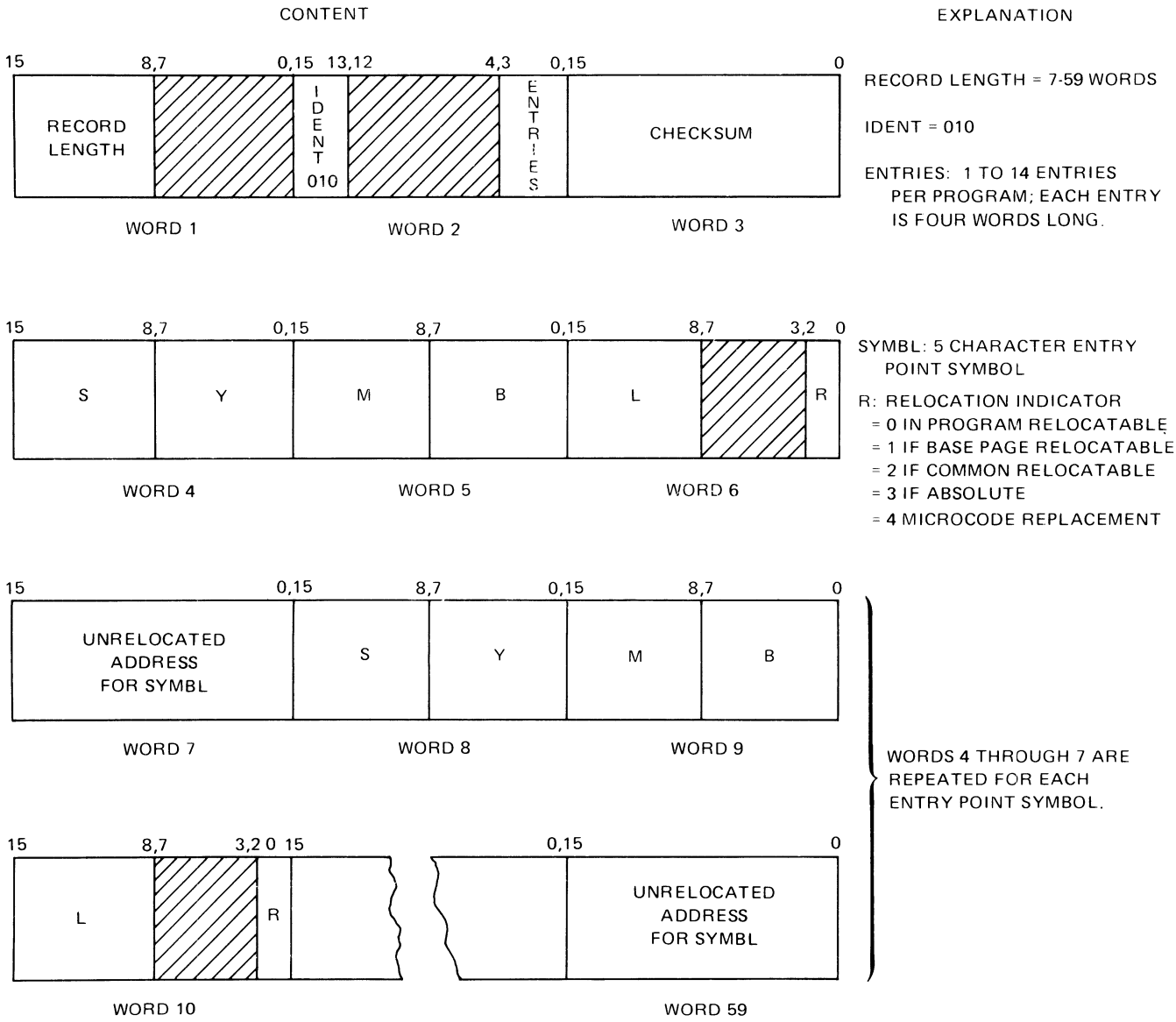
A/C: BINARY TAPE PRECESSION
= 0 IF ASSEMBLER PRODUCED
= 1 IF COMPILER PRODUCED

AMD CURRENTLY USES THE FIRST 15 CHARACTERS OF COMMENTS FOR PART NUMBER AND REVISION CODE. THIS FEATURE IS SUPPORTED BY AN AMD ASSEMBLER.

▨ HATCH-MARKED AREAS SHOULD BE ZERO-FILLED WHEN THE RECORDS ARE GENERATED

TODS—14

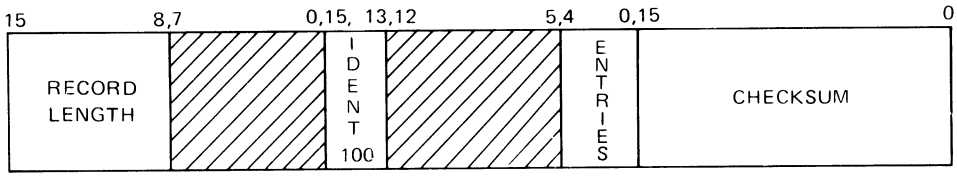
ENT RECORD



TODS—15

EXT RECORD

CONTENT

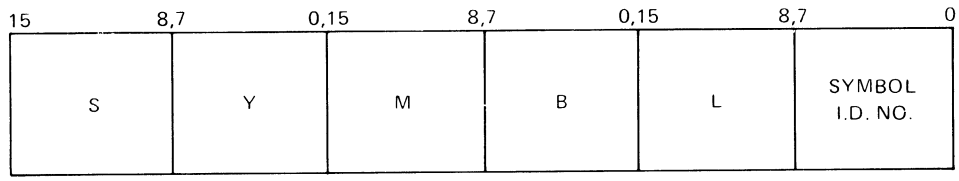


EXPLANATION

RECORD LENGTH = 6-60 WORDS

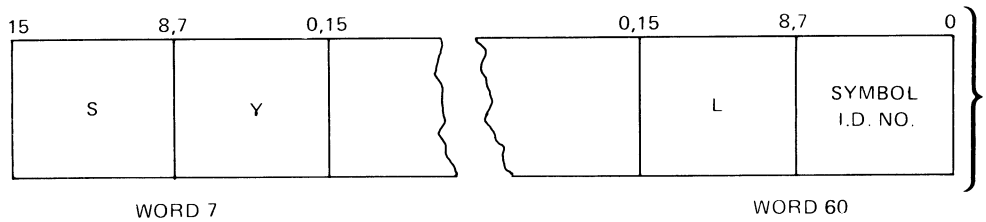
IDENT = 100

ENTRIES: 1 TO 19 PER RECORD; EACH ENTRY IS THREE WORDS LONG



SYMBL: 5 CHARACTER EXTERNAL SYMBOL

SYMBOL ID. NO.: NUMBER ASSIGNED TO SYMBL FOR USE IN LOCATING REFERENCE IN BODY OF PROGRAM.

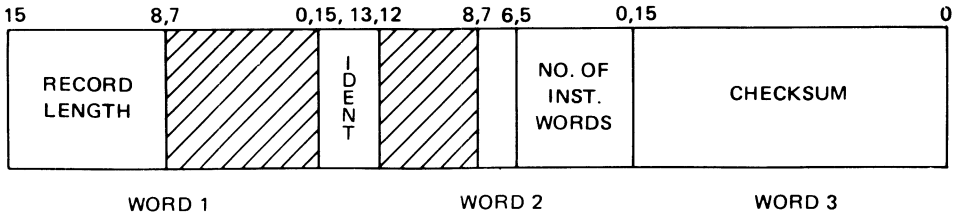


WORDS 4 THROUGH 6 REPEATED FOR EACH EXTERNAL SYMBOL (MAXIMUM OF 19 PER RECORD).

TODS-16

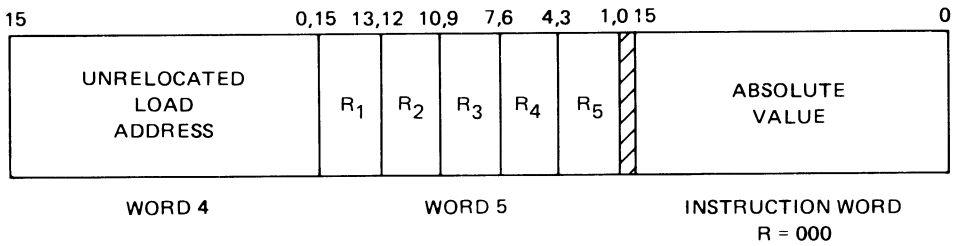
DBL RECORD

CONTENT



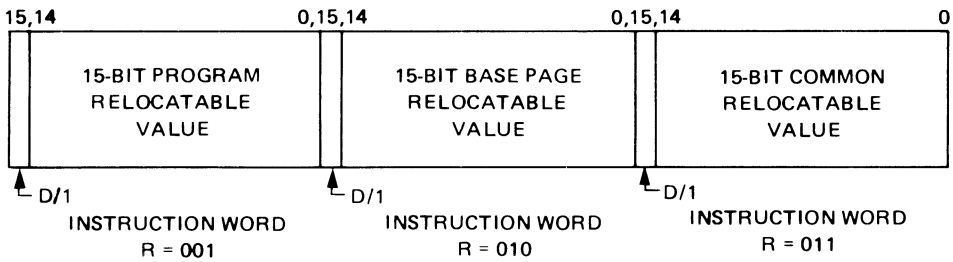
EXPLANATION

RECORD LENGTH = 6-60 WORDS
IDENT = 011
Z/C: RELOCATION OF LOAD ADDRESS
= 0 FOR BASE PAGE
= 1 FOR PROGRAM
= 2 FOR ABSOLUTE
= 3 FOR COMMON
NO. OF INST. WORDS: 1 TO 45 LOADABLE INSTRUCTION WORDS PER RECORD

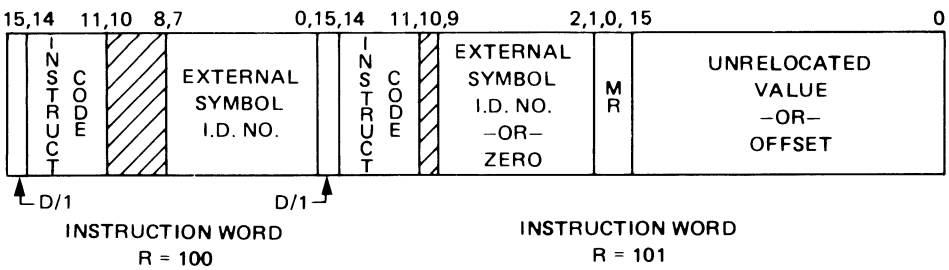


RELOCATABLE LOAD ADDRESS: STARTING ADDRESS FOR LOADING THE INSTRUCTIONS WHICH FOLLOW;

R's: RELOCATION INDICATORS:
000 = ABSOLUTE
001 = 15-BIT PROGRAM RELOCATABLE
010 = 15-BIT BASE PAGE RELOCATABLE
011 = 15-BIT COMMON RELOCATABLE
100 = EXTERNAL REFERENCE
101 = MEMORY REFERENCE



R_1 IS RELOCATION INDICATOR FOR INSTRUCTION $WORD_1$; R_2, FOR INSTRUCTION $WORD_2$; ETC.

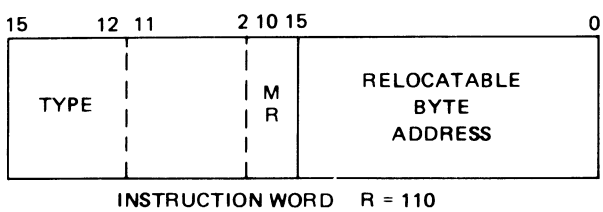


D/I: INDIRECT ADDRESSING

0 = DIRECT
1 = INDIRECT

MEMORY REFERENCE INSTRUCTIONS USE TWO WORDS, WITHIN THE TWO-WORD GROUP, "MR" INDICATES RELOCATABILITY OF OPERAND SPECIFIED IN SECOND WORDS:

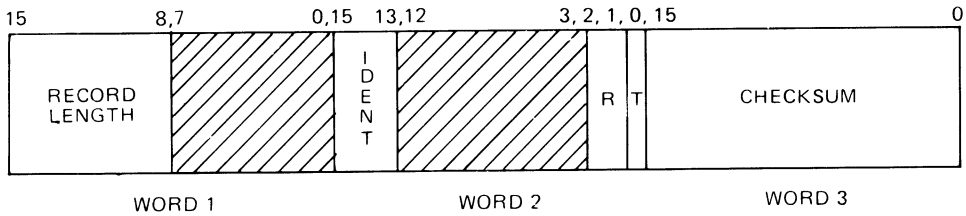
00 = PROGRAM RELOCATABLE
01 = BASE PAGE RELOCATABLE
10 = COMMON RELOCATABLE
11 = ABSOLUTE

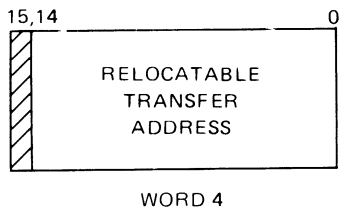


TODS—17

END RECORD

CONTENT





EXPLANATION

RECORD LENGTH = 4 WORDS
IDENT = 101

R: RELOCATION INDICATOR FOR TRANSFER ADDRESS

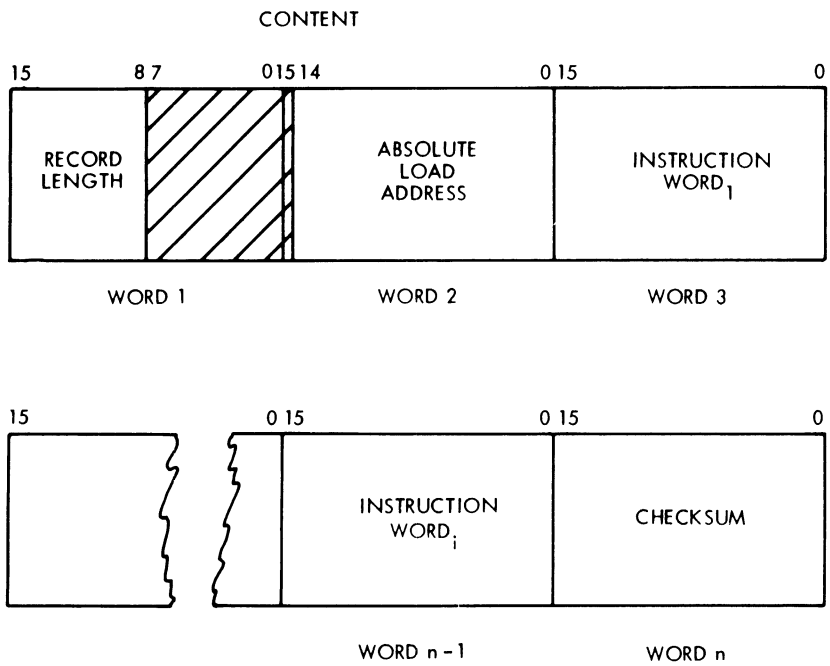
= 0 IF PROGRAM RELOCATABLE
= 1 IF BASE PAGE RELOCATABLE
= 2 IF COMMON RELOCATABLE
= 3 IF ABSOLUTE

T: TRANSFER ADDRESS INDICATOR

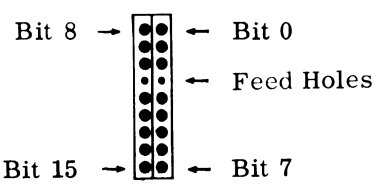
= 0 IF NO TRANSFER ADDRESS IN RECORD
= 1 IF TRANSFER ADDRESS PRESENT

TODS—18

ABSOLUTE TAPE FORMAT



†Each word represents two frames arranged as follows:



APPENDIX I
RTE VS RTE-II

GENERAL

To aid the user in learning some of the new features of the Real-Time Executive II Software System, this Appendix lists some of the differences between the old and new software.

PROGRAM SCHEDULING

In old RTE, assume there exists a "PROGA" & "PROGB".

1. PROGA is scheduled
2. PROGA schedules PROGB with wait.
3. PROGB runs.
4. PROGB schedules itself with offset.
5. PROGA runs, since PROGB went dormant.

In RTE-II, the above process is different.

1. PROGA is scheduled.
2. PROGA schedules PROGB with wait.
3. PROGB runs.
4. PROGB schedules itself with offset.
5. PROGB eventually runs.
6. PROGB terminates.
7. PROGA runs, since PROGB went dormant.

In summary, when PROGA schedules PROGB with wait, PROGA will not be put back into the scheduled list until PROGB has terminated (an EXEC 6 call). Therefore, if the programmer's intention was to schedule with offset and have PROGA run, he will now:

1. Schedule with offset, using PROGA name instead of Ø.
2 Terminate, saving resources.
3. Branch to restart location.

If the programmer's intention was simply to be rescheduled at some unknown future time by PROGA, and all he really wants to do is to save resources and restart from the point of suspension in a swapped copy, he will:

1. Terminate, saving resources.
2. Branch to restart location.

RTE-II DORMANT LIST NON-STRUCTURE

The old RTE's dormant list is priority structured, which means each entry and exit from the dormant list requires a call to "$LIST" as well as a program name search/compare (5 bytes).

RTE-II has replaced the dormant list-by-priority with a scan of all IDS via the keyword block to search for a program scheduled by name, and no search for a program scheduled by address. This technique greatly decreases the time required to move a program into the scheduled list since no dormant list restructuring is required.

DIFFERENCES DUE TO BACKGROUND SWAPPING

Assume:

FMGR	type 3,	priority 1
ASMB	type 3,	priority 2
XREF	type 3,	priority 3
LOADR	type 3,	priority 4

Also, a paper tape has been prepared as follows:

- (FMGR commands and exit)
- (ASMB source program)
- (Program data)

Now, type in the following commands:

```
*ON,FMGR,5
*LG,10
*ON,ASMB,5,99
*ON,LOADR,99
*ON,MYPRG,5
```

In the old system there would be no problem since all background disc resident programs execute to completion; therefore, the information on the paper tape will be read by the proper program at the proper time.

In the RTE-II system, these programs will swap in the background and the paper tape will be read by whoever is in core, the results will be confused.

CHANGES IN BASE PAGE

LOCATION	OLD MEANING	NEW MEANING
001710	Head of dormant list	Not used
001713	Not used	Head of wait list

DEVICE REFERENCE TABLE CHANGES

Old DRT format.

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

(Sub Chan) (EQT Entry Number)

New DRT format.

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

(Sub Chan) (Lock Flag) (EQT Entry Number)

This new DRT format permits:

1. LU locking by up to 31 programs.
2. Increased subchannel capability to decimal 32.

TIME KEEPING

Old system Time-of-Day

WORDS	SYSTEM T-O-D	PROGRAM IDS
1	Day of year	
1	Hour of day	Hour to run
1	Min of hour	Min to run
1	Sec of hour	Sec to run
1	10ms of sec	10ms to run
TOTAL = 5 words		4 words per ID

NOTE

The systems 5-word T-O-D is updated each 10ms Time Base Generator tick.

New system Time-of-Day

WORDS	SYSTEM T-O-D	PROGRAM IDS
1	Day of Year/Year	
2	T-O-D -24 hours	Time-to-Run -24 hours
TOTAL = 3 words		2 words per ID

NOTE: Advantages:

1. More efficient time maintenance
2. Gained year to assist in calculating month, day
3. 2 less word in system
4. 2 less words per ID

INDEX

INDEX (Continued)

INDEX (Continued)

INDEX (Continued)

INDEX (Continued)

INDEX (Continued)

HEWLETT hp PACKARD

MANUAL PART NO.

92001-93001

PRINTED IN U.S.A.