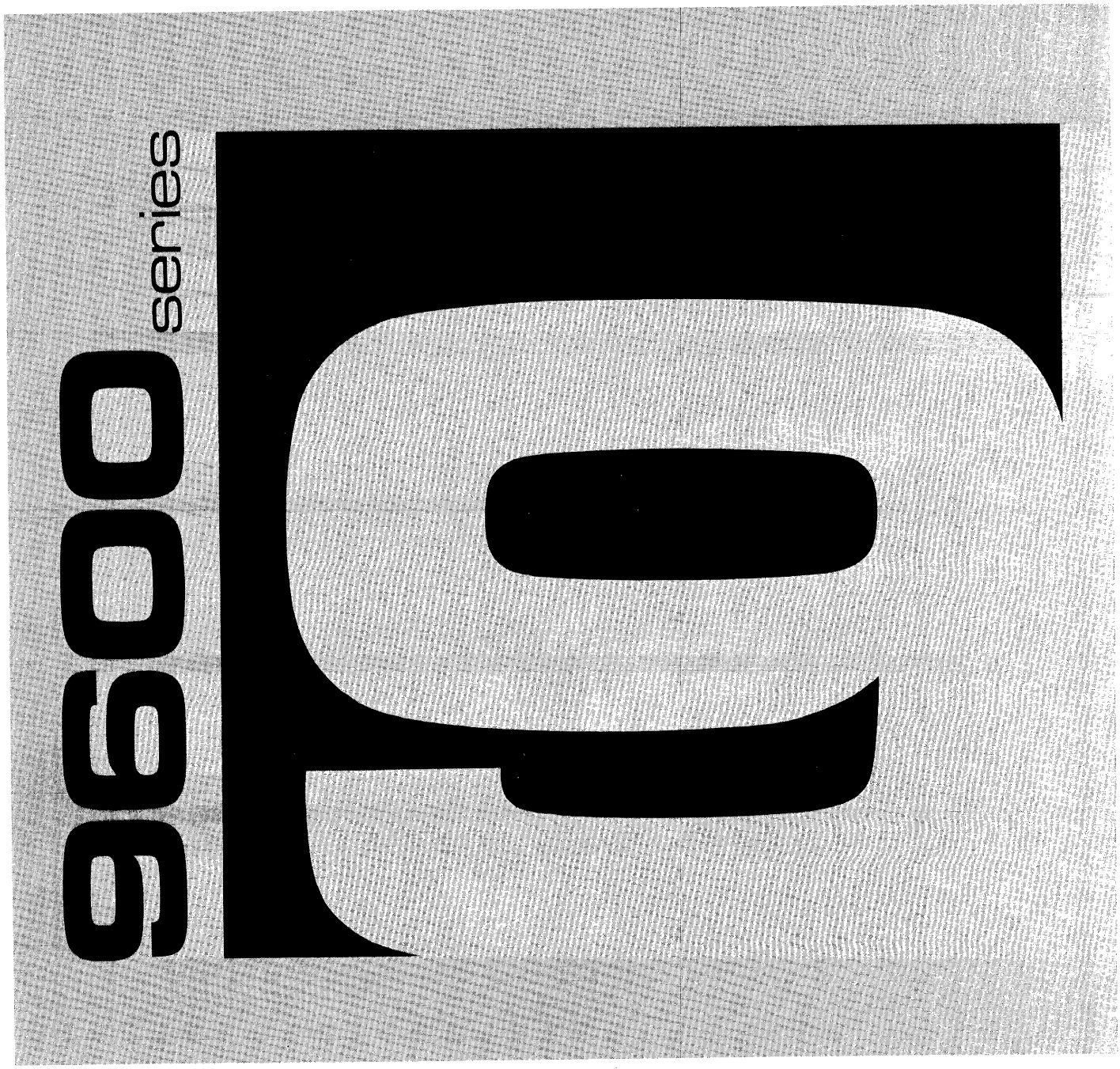
HEWLETT hp PACKARD

Real-Time Executive III Software System

Programming and Operating Manual



Real-Time Executive III Software System

Programming and Operating Manual

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

PART NO. 92060-90004

Printed in U.S.A. 7/76

LIST OF EFFECTIVE PAGES

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 Insert latest changed pages and destroy superseded pages.

Change 0 (Original) . . . . . Jul 1976

DOCUMENTATION MAP

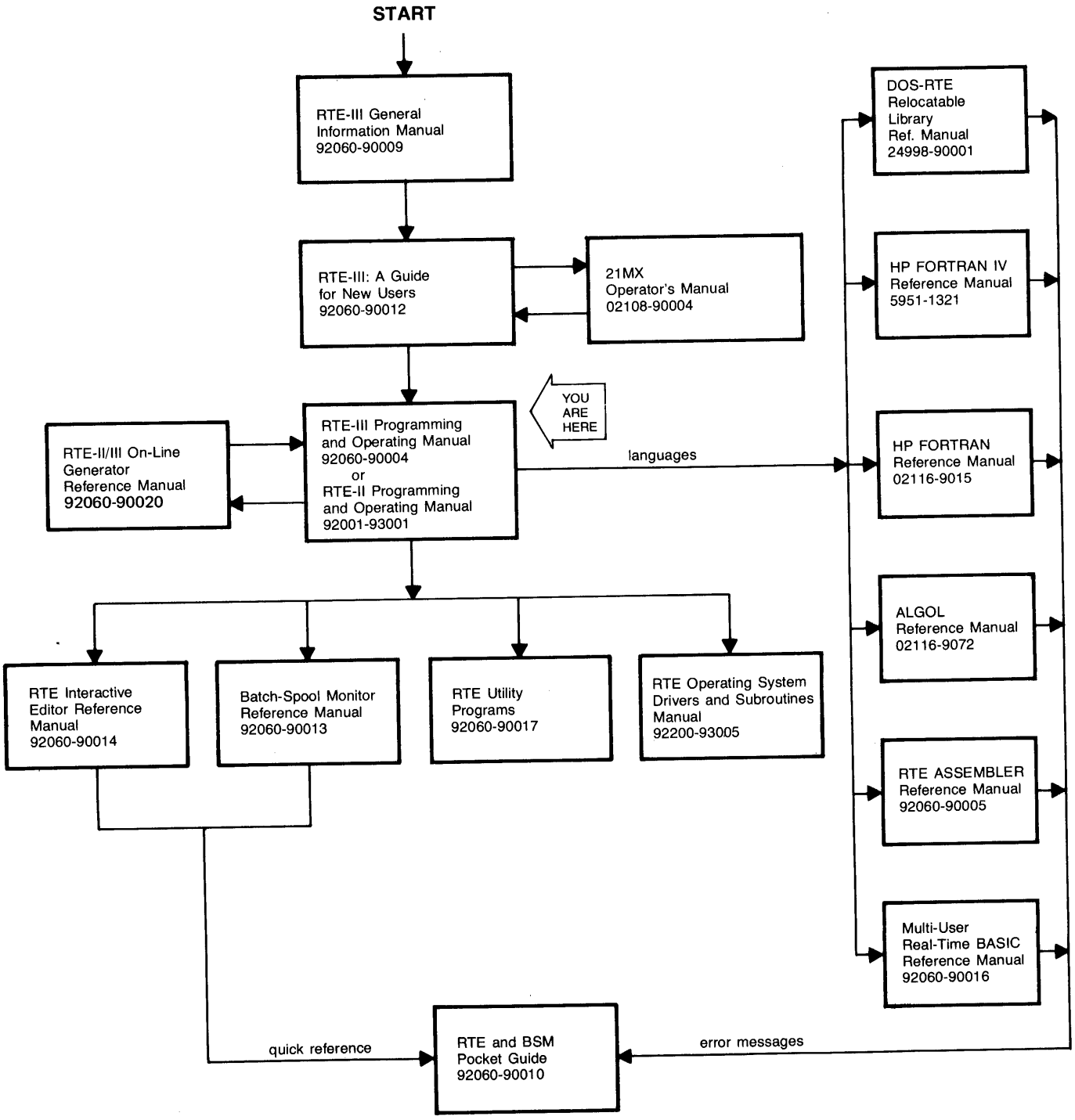


TABLE OF CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

GLOSSARY OF TERMS USED IN THIS MANUAL

ABSOLUTE SYSTEM – The binary memory image of the Real-Time Executive III System (stored on logical unit 2).

AUXILIARY SUBCHANNEL – The subchannel is optional and when used is assigned to logical unit 3. (The binary memory image of RTE-III does not reside on the auxiliary subchannel.) The auxiliary subchannel has the same status as the system subchannel in that it is treated as a logical extension of the system subchannel.

CLASS I/O – A method of communication between a set of programs or devices that may be synchronous or asynchronous with respect to each other, in order to provide parallel processing of information. Class I/O allows a program to continue processing after initiating the operation, without requiring that it wait for completion (I/O without wait).

DEVICE DOWN – Relates to the state of a peripheral I/O controller or device. When the controller or device is down, it is no longer operable. Also, refers to the operator command DN, which sets the controller or device down.

DEVICE UP – Relates to the state of a peripheral I/O controller or device. When the controller or device is up, it is operable. Also, refers to the operator command UP, which sets the controller and all associated devices up after being set down.

EQT (EQUIPMENT TABLE) – A table in memory associating each I/O interrupt location (I/O controller) with a particular software processing routine (driver). The status of the I/O controller and information about any current request is also stored in its EQT.

GLOBAL TRACKS – Global tracks are a subset of system tracks and are accounted for in the track assignment table. Any program can read/write or release a global track (i.e., programs can share global tracks).

I/O CONTROLLER – A combination of I/O card, cable, and (for some devices) controller box used to control one or more I/O devices on a channel.

I/O DEVICE – A physical unit defined by an EQT entry (I/O controller) and subchannel.

LG AREA – A group of tracks used to temporarily store the relocatable output of an assembler, compiler, or file manager prior to relocation by the loader.

LOGICAL MEMORY – Logical memory is the 32K (maximum) address space described by the currently enabled memory map. If the System Map is enabled, it describes those areas of physical memory necessary for the operation of the operating system and does not change during system operation. When the User Map is enabled, it is updated to describe those areas needed by programs when it is to be executed. DMA Maps describe buffers during DMA transfers.

LU (LOGICAL UNIT) NUMBER – A number used by a program to refer to an I/O device. Programs do not refer directly to the physical I/O device channel number, but through the LU number which has a cross reference to the device. This allows I/O devices to be changed without having to change the programs.

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.

PARTITION – A block of memory with a fixed size (in pages) and identification number located in the disc resident program area. The user may divide up the disc resident program area into as many as 64 partitions classified as a mixture of Real-Time and Background, all Real-Time, or all Background. Disc resident programs run in partitions.

PERIPHERAL SUBCHANNEL – Peripheral subchannel is a subchannel that is available to the user for read/write operations but for which RTE-III does not manage the subchannel nor maintain a track assignment table. (The file manager can, however, use peripheral subchannel tracks.) A peripheral subchannel must have a logical unit number assignment greater than 6.

PHYSICAL MEMORY – Physical memory is all memory available to the user. Physical memory includes the operating system, libraries, common, system available memory, and all partitions.

PROGRAM SWAPPING – Where disc resident program A is removed from a partition and stored on the disc in its current state of execution, and program B is placed (for execution) in the partition formerly occupied by program A. Program A is eventually returned to either the same or a different partition to continue.

REAL-TIME EXECUTIVE III – The total operating system comprised of the memory resident modules (e.g., EXEC,

SCHED, RTIOC), plus I/O drivers, and various tables. Abbreviated RTE-III.

RESOURCE MANAGEMENT – Resource management, or numbering, is a feature that allows the user to manage a specific resource shared by a particular set of programs, so that no two of these programs use the resource at the same time.

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

SUBCHANNEL – One of a group of I/O devices connected to a single I/O controller. For example, RTE driver DVRxx can operate more than one magnetic tape drive through subchannel assignments. In the case of moving head discs, contiguous groups of tracks are treated as separated subchannels. For example, a 7905 disc platter may be divided into four subchannels.

SYSTEM SUBCHANNEL – The disc subchannel assigned to logical unit 2 that contains the binary memory image of the Real-Time Executive III System.

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

TIME-OUT – 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 RTE-III will wait for the device to respond to an I/O transfer command before RTE-III makes the device inoperable.

SECTION I
GENERAL DESCRIPTION

INTRODUCTION

The Hewlett-Packard Real-Time Executive with dynamic mapping (RTE-III) is a multiprogramming operating system that supports user access to more than one million words of main memory. Through a unique scheme of memory management and mapping, the central processor unit (CPU) can be expanded from a maximum of 32K words of "logical" memory to 1024K words of "physical" memory. "Logical" memory describes the actual 32K address space imposed by the 15-bit address length used in HP 2100 Series Computers. Note that 32K is also the minimum amount of memory that RTE-III will operate in. The term "Physical" memory describes all of memory available to the user through the memory management and mapping scheme. The RTE-III system takes care of all addressing and mapping for you (except in the case of privileged drivers which are discussed in Section V). With the exception of those features listed in Appendix I, RTE-III is backward compatible with RTE-II. Most programs written for a previous HP Real-Time executive system, and using documented system functions, will operate correctly under RTE-III.

This book serves as a reference manual and programmer's guide to the RTE-III system. You should be familiar with operating procedures of the HP 21MX Series Computers and other system related hardware, and in addition, with software programming Languages as presented in the Hewlett-Packard FORTRAN (02116-9015), FORTRAN IV (5951-1321), ALGOL (2116-9072), and 21MX Assembler (24307-90014) Programmer's Reference Manuals.

HARDWARE

The RTE-III system operates with the minimum hardware configuration shown in Table 1-1.

SOFTWARE

The RTE-III software is listed in *92060B Software Numbering Catalog*, part number 92060-90019.

Table 1-1. Minimum RTE-III System

HP 21MX Computer with 32K Memory
Time Base Generator
Dual Channel Port Controller (DCPC)*
Dynamic Mapping System
Memory Protect
System Console Device
High Speed Disc Storage
HP Mini Cartridge Subsystem or High Speed Paper Tape Reader
*Note that DCPC provides the direct memory access (DMA) capability.

MEMORY MANAGEMENT

The Real-Time Executive, using the Dynamic Mapping System, provides the capability of addressing memory configurations larger than 32K. Using DMS, the user has the capability of addressing up to 1024K words of physical memory.

The capability of addressing more than 32K is accomplished by translating memory addresses through one of four "memory maps"; a memory map being defined as 32 hardware registers that provide the interface between the 32K logical memory and physical memory.

Note that all memory map addressing is done internally by the system and is transparent to the user. The following brief explanation of the addressing and mapping process provides a general understanding of the overall operation of the system. For a more detailed description of the Dynamic Mapping System, refer to the 21MX Computer Reference Manual, HP Part No. 02108-90002.

ADDRESSING

The basic addressing scheme of the computer uses a 15-bit number which describes a location in memory numbered

0 to 32767 (refer to Figure 1-1). The 32768 (32K) locations are grouped into 32 pages, each page containing 1024 (1K) words. The scheme takes the 15-bit address and splits it into two parts. The upper 5 bits (bits 10-14), become the logical page number, an index pointing to one of the 32 registers within a memory map (only one of the four maps can be enabled at a time). The lower 10 bits point to a relative address (or offset) within the destination page and do not require translation. Thus, when the address is converted, the index is used to determine which of the 32 registers of the currently enabled map has the 10 bit physical page address. This page address is then concatenated to the relative address to provide the ultimate 20 bit address in physical memory.

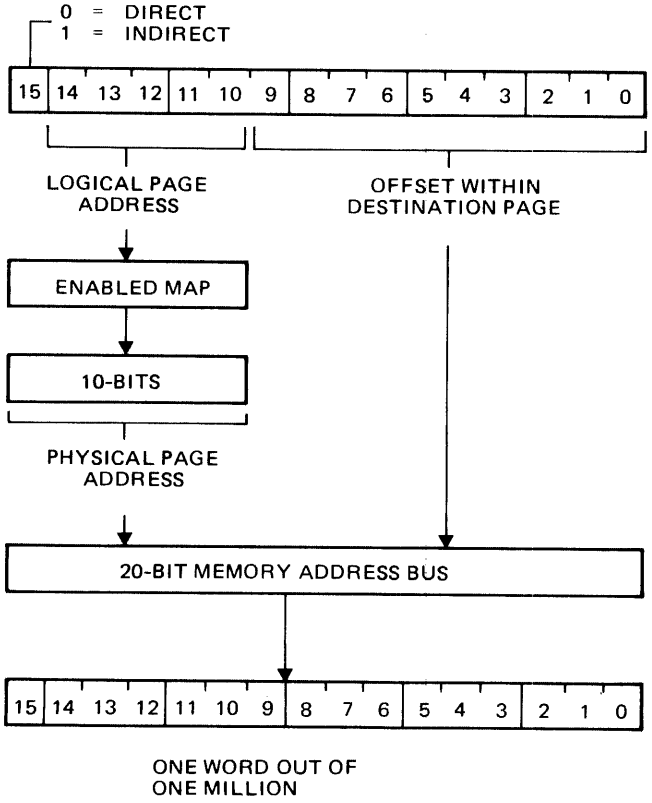


Figure 1-1. RTE-III Address Scheme

MAPS

There are four memory maps managed by the system – the User Map for describing user programs, the System Map for describing the system and system available memory, and two Dual Channel Port Controller (DCPC) maps called Port A Map and Port B Map for describing the initiator of the direct memory access (DMA) transfer. At any one instant only one memory map is enabled. This map describes the 32K logical address space at that instant in time. Either the System or User Map will be enabled unless a DMA transfer is in progress. If a DMA transfer is taking place, the appropriate Port Map is instantaneously enabled each time a word is transferred between the DCPC and main memory.

SYSTEM MAP. This map, which is automatically enabled whenever an interrupt occurs, is loaded by the system during system initialization and is never changed. It describes the logical address space which includes the RTE-III system and its base page, the memory resident library, and system available memory. The inclusion of common in the system map is a generation option for the benefit of users with privileged drivers. Refer to Section V for more information on privileged drivers.

USER MAP. Associated with each disc resident program is a unique set of page addresses that describe the logical address space for that program. These page addresses describe the memory occupied by the system, the memory resident library, common (if the program uses it), the program's base page, and the program. All memory resident programs use a single set of page addresses describing the system base page, the system, the memory resident library, common, and the memory resident program area. Each time a new memory or disc resident program is dispatched, the system reloads the User Map with the appropriate set of page addresses. The User Map therefore, provides the interface between logical memory and physical memory.

PORT A MAP. Direct memory access is a software assignable direct data path between memory and a high speed peripheral device. This function is provided by the 21MX Dual Channel Port Controller (DCPC). There are two DCPC channels, each of which may be assigned to operate with an I/O device. The Port A Map is automatically enabled when a transfer on DCPC channel one takes place. It must be reloaded by the system each time the channel is assigned so that the caller's buffer is described. Having separate maps for DCPC facilitates multi-programming since DCPC may be accessing one program's buffer while another program (in a different area of physical memory) is using the CPU under the User Map (i.e., when one program is using DCPC, another program can be executing).

PORT B MAP. This map is handled the same as the Port A Map except that it applies to DCPC channel two.

PHYSICAL MEMORY

At generation time, the user plans the physical memory

allocations shown in Figure 1-2 and then loads the system components in the most efficient configuration. The user determines the size of system available memory, the number and size of each partition, the size of common, and the size and composition of the resident library and memory resident program area.

In Figure 1-2 the areas shown are used as follows:

- System Base Page – contains system communication area and is used by the system to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, etc. System and library links, memory resident program links, and trap cells are also located on the system base page. The base page links for memory resident programs are not accessible by disc resident programs and therefore may not be shared. System and library links and the system communication area are accessible to all programs. Partition base pages, used for disc resident program links, are described below with partitions.

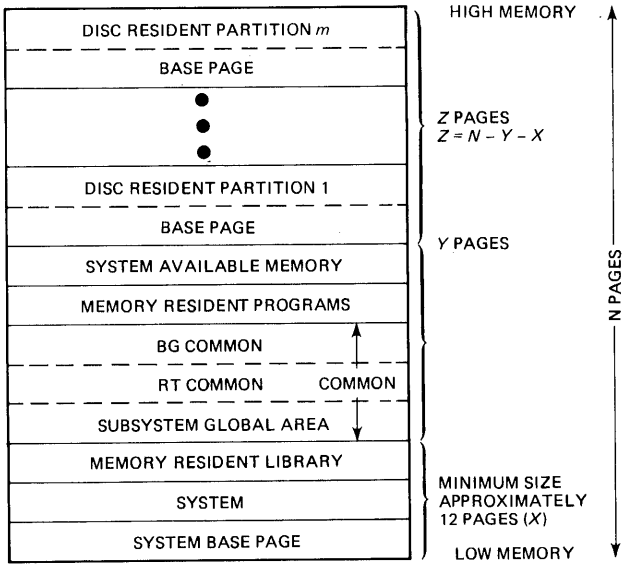


Figure 1-2. Physical Memory Allocations

- System – Contains the absolute code of the type ∅ system modules (e.g. RTIOC, SCHED, EXEC), and drivers plus tables that form the nucleus of the operating system.

- Memory Resident Library – Contains those re-entrant or privileged library routines which are used by the memory resident programs (type 6) or which are force-loaded (type 14) at generation time.

- Common – This area is divided into three subareas: The Subsystem Global Area (SSGA), the Real-time Common area, and the Background Common area. SSGA is used by Hewlett-Packard software subsystems for buffering and communications. The Real-time and Background subareas are reserved for user-written programs that declare COMMON.

- Memory Resident Programs – This area contains all type 1 programs that were loaded during generation.

- System Available Memory – This is a temporary storage area used by the system for Class I/O and reentrant I/O (refer to Section III), and automatic buffering.

- Partition – This is an area set aside by the user where his disc resident program will run. Each partition has its own base page that describes the linkages for the program running in the partition. Up to 64 partitions are allowed, subject to available main memory.

All of the above areas are established during system generation. Refer to Section VI for more information on sizes and boundary determination.

LOGICAL MEMORY

Logical memory is the 32K (maximum) address space described by the currently enabled memory map. If the System Map is enabled, logical memory includes the operating system and its base page, the memory resident library, and system available memory. It may also include common if that option was chosen at generation time. If the User Map is enabled for a disc resident program, logical memory includes the operating system, the memory resident library, common (if it is used by the program), and the currently executing program and its base page. The logical memory of a memory resident program includes the system base page, the operating system, the library, common and all memory resident programs. Port Maps are enabled on a word-at-a-time basis during DCPC transfers. They describe the logical memory containing a data buffer. A Port Map will be the same as either the System Map or the map of the program being serviced, depending on type of I/O call.

Figure 1-3 shows four configurations of the 32K logical address space. The first configuration is how this space appears under control of the System Map. Note that there is always a total of 32 pages to be divided up; however, the particular boundaries shown for the various parts are only examples. Your system could be larger or smaller. The

second configuration is how the logical address space appears under control of the User Map when a memory resident program is executing. The third configuration is how the logical address space appears under control of the User Map when a disc resident program using common is executing. The fourth configuration is how the logical address space appears under control of the User Map when a disc resident program not using common is executing. Many programs will not require a full 32K space, and unneeded pages will be READ/WRITE protected in the user map (shown in Figure 1-3, configuration 3).

SYSTEM DESCRIPTION

MULTIPROGRAMMING

The RTE-III System is a multiprogramming system that allows several programs written in Real-Time Assembler, ALGOL, or FORTRAN languages, to operate concurrently, each program executing during the unused central processor time of the others.

Up to 256 programs may be defined by ID segments at one time. An ID segment is a table that describes the program. Refer to Appendix A for more information. Note that additional programs may be relocated and then saved by the file manager. This increases the number of readily accessible programs indefinitely.

RTE-III has a scheduling module that decides when to execute the competing programs, which may be scheduled by time intervals, an external event, an operator request, or by another program. All input/output and interrupt processing is controlled by RTE-III except for privileged interrupts which circumvent the system for a faster response. When a program requests a nonbuffered I/O transfer, the system places the program in an I/O suspended state, initiates the I/O operation, and starts executing the next highest priority scheduled program. When the I/O transfer is complete, the system reschedules the suspended program for execution. When a program requests a buffered I/O transfer, the system does not suspend the program.

There are two areas available to the user for execution of his programs, the memory resident program area and the disc resident program area. Note that the number of programs which may be resident in memory at one time is the sum of the number of memory resident programs and the number of partitions defined (up to 64). This greatly minimizes the dispatching time for disc resident programs which are in memory, because it is much faster to switch maps than it is to perform a swap between main memory and disc.

MEMORY RESIDENT PROGRAM AREA

The memory resident program area is that area where programs are always resident and is intended for high priority tasks requiring quick response time to real-time conditions and for often used programs which are very small.

DISC RESIDENT PROGRAM AREA (PARTITIONS)

The disc resident program area is divided up into partitions which are defined as fixed blocks in memory established at generation time. The user may specify how many partitions (up to 64) are in the system, what type each partition is (Real-Time or Background), and the size (pages) of each partition.

The number of partitions depends on the amount of physical memory available. Partition types can be specified as a mixture of real-time and background, all real-time, or all background. A program can be assigned at load time to run in any partition that is large enough to accommodate it. There can be several programs assigned to the same partition, but only one program can run in that partition at a time. If a program is not assigned to a partition, then by default, real-time programs will run in real-time partitions and background programs in background partitions. If only one type of partition is defined, all programs will run in that type partition.

When a program is not assigned to a partition, it will run in the smallest free partition that will accommodate it. If no free partition exists, the program will be swapped with the lowest priority swappable program that occupies a partition large enough to accommodate the new program.

BASE PAGE

In RTE-III, the system, and each disc resident program has its own base page. The system base page contains the system communication area, system and library links, memory resident program links and trap cells for interrupt processing. The disc resident program base page contains the system communication area, system and library links, and disc resident program links. The base page communications area (see Appendix A) and the system and resident library links, which are located in physical page 0, will be common to all base pages.

Instructions are channeled to the proper physical base page by being either above or below a base page fence set during generation. The base page fence not to be confused with the memory protect fence, is set to be immediately below the communication area, the system links and the library links. If a base page address is above the fence it is

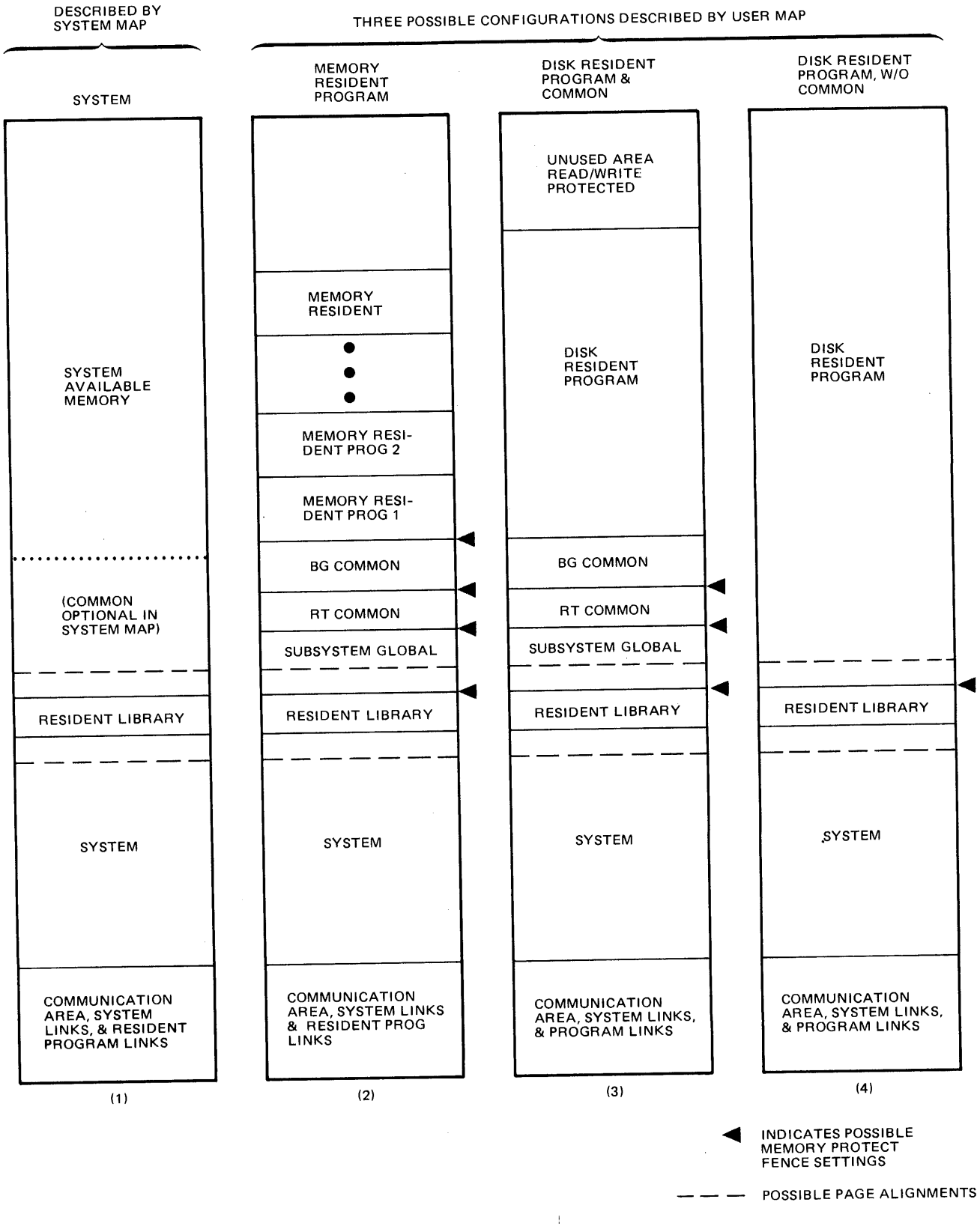


Figure 1-3. RTE-III 32K Logical Memory Configurations

channeled to physical page 0. If it is below the fence it is channeled to the base page described by the currently enabled map.

COMMON

The real-time and background commons along with subsystem global area occupy a contiguous area in memory and are treated as a single group for mapping purposes (refer to Figure 1-3). The use of common is optional on a program basis; that is, any program may use real-time common, background common or no common. If the program declares common, both common areas and the subsystem global area will be included in the User Map. If the program does not use common it is not accounted for in the User Map, thereby providing the user a larger program area in the 32K logical address space. Note that memory resident programs always include common in their map.

REAL-TIME AND BACKGROUND COMMON. If a program declares at least one word of common, the use of real-time or background common is selected by program type (at generation) or parameters with the on-line loader. Program types are summarized in Appendix I. Note that the memory protect fence protects areas below the selected common.

These system common areas are not to be confused with the local common area which may be specified for programs loaded on-line. The system common areas are sharable by programs operating in different partitions, whereas the local common area is appended to the program (i.e. it will be in its partition) and is accessible only to that program, its subroutines, and its segments.

SUBSYSTEM GLOBAL AREA. The subsystem global area consists of all type 30 modules input to the generator. Accessed by entry point (using EXT statements) rather than common declarations, SSGA provides multiple communication and buffer areas for Hewlett-Packard subsystems. SSGA access is authorized by program type at generation (see Appendix I) or through special parameters to the on-line loader. Programs authorized for SSGA access include the common area in their maps and have the memory protect fence set below SSGA.

MEMORY PROTECTION

Memory protection between disc resident program partitions and between disc and memory resident programs is provided by the Dynamic Mapping System. A program cannot access a page not included in its logical memory either directly or through a DMA transfer. Since many programs do not use all of the possible 32K logical area, unused logical pages above the program are READ/WRITE protected and do not necessarily have counterparts in physical memory.

A different form of protection is required for the system, library, and (optionally), common. The memory protect fence provides this protection by preventing stores and jumps to locations below a specified address. All possible fence positions are shown in Figure 1-3.

The memory protect fence applies to the logical address space where addresses are compared to the fence before translation. If a disc resident program does not use any of the common areas, the memory protect fence is set at the bottom of the program area. Similarly, for a memory resident program not using common, the memory protect fence is set at the base of the entire memory resident area.

For programs using common, all of logical memory including common is mapped and the fence is set at one of three possible locations, depending on the portion of common being used. A hierarchy of protection is thereby established within common due to their physical locations. Background common is the least protected (any program using any common can modify it) and SSGA is the most protected (only programs authorized for SSGA access can modify it). Figure 1-4 expands the common area and shows these three fence settings as (a), (b), and (c).

Figure 1-4 also shows a potential problem area marked "?" which includes those words from the top of common to the next page boundary. This area could include one or more memory resident programs and/or part of System Available Memory. Any program using common could potentially destroy the contents of this area. Aligning the top of common at the next page boundary is a generation option that expands the size of background common while eliminating

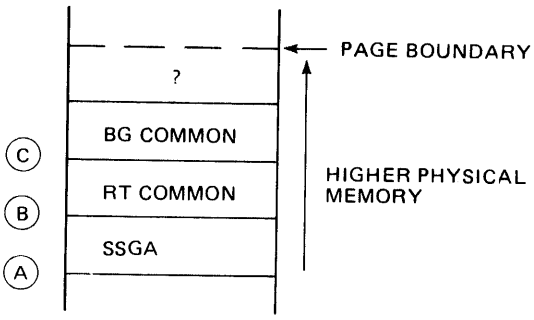


Figure 1-4. Memory Protect Fence Locations for Programs using Common.

this problem. The same option is available for the boundary between memory resident programs and system available memory where a similiar problem occurs.

PROGRAM TYPES

As described above, there are two areas for program execution, the memory resident area and the disc resident area.

MEMORY RESIDENT PROGRAMS

The memory resident area is intended for high priority tasks requiring rapid response to real-time conditions and external events and for very small programs which execute very often. Memory resident programs are loaded into the system during generation.

DISC RESIDENT PROGRAMS

Disc resident programs run in partitions and require more time for dispatching than memory resident programs do since a partition must be allocated and the program loaded from the disc. There are two types of programs that run in the partitions, real-time programs and background programs. To minimize the time necessary to locate a partition for a particular program the capabilities for defining two types of partitions and for assigning programs to partitions have been included in the system. These capabilities minimize the number of programs competing for the same partitions. The real-time programs are usually user programs that have been compiled to handle external events. These new programs can be compiled, tested, and placed into operation without any paper tape output and little operator intervention. Background disc resident programs are usually compilers, assemblers, and editors used to create new programs to control future processes while the real-time programs handle external events. Background disc resident software usually includes the following items.

SYSTEM COMMAND PROGRAM ($$CMD). A system program scheduled by SCHED for the processing of the LU, EQ, and TO system commands.

ASSEMBLER. Accepts source programs in HP Assembler Language and outputs a relocatable binary program to LG disc tracks and/or punches the program on paper tape.

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

FORTRAN IV. Compiles source programs written in FORTRAN IV with extended precision arithmetic and bit manipulation via logical expressions.

ALGOL. Compiles source programs written in HP ALGOL.

INTERACTIVE EDITOR (EDITR). EDITR is an on-line editor that provides the user with a powerful editing tool. 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.

RELOCATING LOADER. Provides on-line loading of user generated programs. Programs can be debugged and tested in background and then loaded to run in either a real-time or disc resident partition.

ON-LINE GENERATOR. Provides a way to configure a new operating system on-line under control of your current operating system.

SWTCH. Transfers a new operating system from a file created by the On-Line Generator to a disc subchannel.

DEBUG. When user program is loaded with relocating loader, DEBUG can be appended to each main program and segment to provide checkout. When the program is run, DEBUG takes control of program execution and requests instructions from the keyboard.

DISC RESIDENT PROGRAM SIZE

Disc resident programs run in partitions and partitions always begin on a page boundary. Referring to Figure 1-3, it can be seen that the program (starting with the second page of the partition) is placed in logical memory immediately following the background system common if the program uses common, or immediately following the library if not. When the program runs, the user program only sees that area of memory described by its map.

The end of logical memory is determined by the number of pages needed to hold the main program, its subroutines, and its largest overlayable segment. This, plus a base page, also determines the minimum acceptable partition size for the program. If a program needs a dynamic buffer area or symbol table space (EDITR, for example), the program size in pages must be increased during generation or when loading with the relocating loader to make it appear larger. This insures enough buffer space for the program to use, since only this number of pages will be mapped into the program area of the logical address space. If the partition allocated is larger than the program, the remaining pages are not a part of its logical memory, and are therefore inaccessible. This is necessary to allow a program to be freely swapped between disc and any suitable partition during its operation.

SUBROUTINES

Each user program (main or segment) consists of a primary routine, containing the transfer point for entry into the program from RTE-III and optionally a series of subroutines. In Assembly Language, the transfer point is given by the label appearing as the operand of 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 – Code is shared and the routine can be interrupted.
- Privileged – Cannot be interrupted except by privileged I/O.
- Utility – Separate copy of the routine must be appended to each calling program.

The classification of a specific routine is based on its function, word length, and execution time. A single copy of a re-entrant or privileged subroutine may be used by more than one program if it is included in the resident library. (See Figure 1-2.) If called by a disc resident program and not in the resident library, a copy of the re-entrant or privileged subroutine is appended to the calling program during relocation.

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 memory 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). Note that utility subroutines may call a re-entrant or privileged subroutine but the reverse is not permitted.

The classification into which a particular subroutine fits is determined by program type at generation (See Appendix I).

PROGRAM SCHEDULING

Scheduling of all programs is done by a scheduling module in the memory resident system, 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. RTE-III 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, the system decides in favor of the highest priority program. Priorities are assigned by the user during generation or on-line loading, and may be changed by an operator request.

In RTE-III, the highest priority program scheduled for execution executes first. Then, if that program suspends execution (e.g., waiting for I/O to complete), the next highest priority scheduled program executes.

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
- Waiting for a logical unit number to be unlocked
- Waiting for a resource number to be unlocked
- Waiting for a class number

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. Time scheduling is based on multiples of hours, minutes, seconds, and tens of milliseconds.

PRIORITY LEVEL

Program priority determines the order of a program in the scheduled and I/O 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-III 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 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

Program initiation requires that a copy of the program be present in main memory and that the ID segment be placed in the scheduled list. A disc resident program must be transferred into memory while a memory resident program is always immediately available. When a disc resident program is scheduled, a partition is selected by the system. If an uncompleted program already occupies that partition, the uncompleted program is transferred out to the disc and saved in its uncompleted and modified state, the new program is transferred into its place. This operation is called swapping. During the swap a check is made to see if work can be done by another program already residing in main memory. I/O operatons continue concurrently.

Multiple-partitions combined with background swapping make it possible for multiple users to take advantage of the program development facilities of the RTE-III system. For example, one person can be editing a program in one partition while another is entering, compiling, assembling, or loading a program in another partition. Multiple partitions greatly decrease the amount of time-consuming swapping. Unless assigned to a specific partition, there is no requirement that a swapped-out program be swapped back into the same partition.

INPUT/OUTPUT CONTROL

A system module called RTIOC is responsible for processing normal system interrupts (non-privileged) and input/output operations. Section V describes the I/O structure of the RTE-III system in detail, emphasizing I/O drivers, and privileged interrupt processing.

INTERRUPT PROCESSING

All interrupts, except privileged interrupts and power fail, cause a transfer to 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, containing entries ordered by hardware interrupt priority, indicates 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 entered directly by RTIOC. Processors that respond to user-controlled devices or interrupt sources are normal user programs and are placed in the scheduled list and executed according to program priority.

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 (JSB, I).

PRIVILEGED INTERRUPT

RTE-III offers a special privileged interrupt feature, using an optional privileged interrupt control card and the hardware priority structure of the computer. A privileged interrupt by-passes normal interrupt processing to achieve faster response for interrupts having the greatest urgency. Privileged drivers must save and restore the state of the machine. See Section V for details.

INPUT/OUTPUT PROCESSING

RTIOC allocates DCPC channels for I/O devices requiring direct memory access (DMA), provides for referencing I/O devices by logical unit number (rather than directly by equipment table entry number of the I/O channel), queues 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. All input/output operations occur concurrently with program execution; one program is executed while others receive I/O service. If a transfer is unbuffered, the requesting program is suspended and the next lower priority scheduled program is executed during the suspension. Class and buffered transfers allow the immediate continuation of the requesting program.

CLASS INPUT/OUTPUT OPERATIONS

Under class I/O, input-output operations for a program are performed concurrently with its execution. This unique scheme within the RTE-III system also allows program-to-program communication and program-to-multiple-device operation. The term "class" as used in this context is likened to an account which is owned by one program but 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 one (if available) to the calling program when a class I/O call is made. Once the number has been allocated, the program can keep it as long as desired and use it to make multiple class I/O calls. When the program is finished with the number it should be returned to the system for use by some other programs.

LOGICAL UNIT LOCK

The RTE-III 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-III, 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.

EXECUTIVE COMMUNICATION

When an executing program makes an EXEC call, it attempts to execute a jump subroutine (JSB) to that portion of the system located in the protected area of memory. This causes a memory protect violation interrupt which is duly processed by the system. The parameters associated with the EXEC call in the calling program are examined. If the parameters are legal, the system processes the request.

Using EXEC calls, which are the line of communication between an executing program and RTE-III, 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
- Recover scheduling strings
- Obtain the time of day
- Set execution time cycles
- Obtain status information on partitions

OPERATOR REQUESTS

The operator retains ultimate control of the RTE-III system with requests entered through the teleprinter keyboard. (See Section II) Operator requests can interrupt RTE-III to:

- Turn programs on and off
- Suspend and restart programs
- Examine the status of any partition, program, or I/O device
- Schedule programs to execute at specified times
- Change the priority of programs
- Set up load-and-go operations and source files
- Declare I/O controllers or 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 memory resident programs and disc resident programs, system programs, library routines, and Real-Time Executive Modules are incorporated into a configured RTE-III System. The RTE-III 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 Off-Line Generator (RTGEN), or the Real-Time On-Line Generator (RT3GN) and SWITCH, the relocatable software modules and user programs are converted into a configured real-time system in memory-image binary format stored on the system disc (LU2). In operation the configured system is loaded (bootstrapped) into the computer from the system 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. System configuration is detailed in Section VI of this manual and the On-Line Generator manual (92060-90016).

SYSTEM/AUXILIARY DISCS

The RTE-III System disc tracks are those for which RTE-III controls and maintains a track usage table. Programs may obtain and release tracks from this area using EXEC calls. 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, load-and-go, and scratch area. They may also be used by user programs for storage.

The main differences between a system disc and an auxiliary disc are:

- The configured system (including the memory resident system, the relocated disc resident programs, and the relocatable library), is stored on the system disc.
- The auxiliary disc is optional.
- Most program swapping takes place on the auxiliary disc.

PERIPHERAL DISCS

Peripheral discs (LU's greater than 6) are not managed by the RTE-III System but can be managed with the file manager program. Track allocation and usage in this case are totally up to the user through the file manager. Note that peripheral disc tracks may be protected the same as those on the system/auxiliary discs.

RTE-III SYSTEM SUMMARY

The Hewlett-Packard Real-Time Executive-III Software System is a multiprogramming, multi-partitioned 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 real-time activities permit. The same computer can also be used by the programming group for ongoing development work with RTE-III'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 while the memory protect fence maintains the integrity of the system area and the Dynamic Mapping System maintains the integrity of other user programs.

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 generation or on-line loading, and may be changed by an operator request.

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

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

Configuration is efficient. System generation may be done off-line or on-line using interactive operator dialog or prebuilt answer files. This results in an operating system configured for a specific hardware system.

SECTION II
OPERATOR REQUESTS

INTRODUCTION

The operator controls an executing Real-Time Executive-III System by operator requests entered through the 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 by pressing any key on the console. When RTE 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-III 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-III 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 controller or 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-III Operator Commands (Continued)

Command Format	Description
ST	Examine the status of programs.
TI	Print the current time.
TM	Set the real-time clock.
TO	Examine and dynamically alter an I/O device's time-out parameter.
UP	Declare I/O controller and associated 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, a system generated clear request is issued to the driver.

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.

Any program with a priority of 1 through 40 will not be suspended for buffer limit so alarm messages, etc., are not inhibited.

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 or ALGOL 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 controller or device down (i.e., unavailable for use by the RTE-III System).

Format:

DN $\begin{matrix} ,eqt \\ ,,lu \end{matrix}$

Where:

eqt	is the EQT entry number of the I/O controller to be set down.
lu	is the LU entry number of the I/O device to be set down.

COMMENTS

Setting an I/O controller (EQT) down effectively sets all devices connected to the I/O channel down by blocking any I/O operations on the channel. The state of the devices (LU's) associated with the channel are unchanged.

Setting the I/O device (LU) down will make only the specific device unavailable. However, all other LU's pointing to the device will also be set down. Other devices using this device's I/O channel are unaffected.

The I/O controller or I/O device is left unavailable until the I/O controller is 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 controller, as recorded in the EQT entry.

Format:

EQ,*eqt*

Where:

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

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.

This command is implemented by the disc resident program $$CMD. If $$CMD is not included in the operating system, the message NO SUCH PROG will be printed when the command is entered.

EQ (buffering)

Purpose:

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

Format:

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

Where:

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

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 during generation.

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,
GOIH, ,*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.

The program may also recover the ASCII command string (up to 80 characters typed after the prompt) that scheduled it by using the String Passage EXEC call (see Section III). If the program was rescheduled with a GOIH (inhibit string passage) or if the program has not suspended itself, the command string is not passed.

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 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:

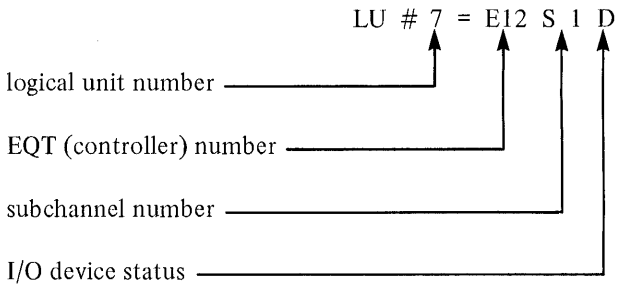
To print the EQT (channel) number and device subchannel number and I/O device status associated with a logical unit number.

Format:

LU,*lu*

Where:

lu	is a logical unit number from 1 to 63.



If the logical unit is disabled (down) then a D is printed as the status; otherwise the position is left blank.

This command is implemented by the disc resident program $$CMD. If $$CMD is not included in the operating system, the message NO SUCH PROG will be printed when the command is entered.

LU (reassignment)

Purpose:

To change a logical unit number assignment.

Format:

LU,*lu*,$\begin{matrix} eqt \\ 0 \end{matrix}$ [,*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 during generation.

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

This command is implemented by the disc resident program $$CMD. If $$CMD is not included in the operating system, the message NO SUCH PROG will be printed when the command is entered.

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-III 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, a system generated clear request is issued to driver.

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 request. 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, a system generated clear request is issued to the driver. 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 5 of Section IV.

ON

Purpose:

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

Format:

ON / ONIH ,*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

The program can recover the ASCII command string (up to 80 characters typed after the prompt) by using the String Passage EXEC call (see Section III). The ONIH command inhibits the passage of the command string.

If the resolution code in the ID segment of the program is not zero, RTE-III 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 the generator or LOADR.

RU

Purpose:

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

Format:

RU
RUIH ,*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

The program can recover the ASCII command string (up to 80 characters typed after the prompt) by using the String Passage EXEC call (see Section III). The RUIH command inhibits the passage of the command string. If there are no characters past *name*, the command string is not transmitted.

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-III 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 and partition number of the program currently occupying memory, or print the name of the program occupying a specified partition.

Format (status of a program):

ST,*name*

Format (name and partition number of current program):

ST,*0*

Format (name of program in specified partition):

ST,*part numb*

Where:

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

0 causes the system to print the name and partition number of the program currently in memory. If none, then 0 is printed.

part numb is a partition number; causes the system to print the name of the program in *part numb.* If the partition is empty, 0 is printed. If *part numb* is wrong, NO SUCH PROG is 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.

When the format ST,0 is used, the status is printed as:

name part numb

Where

name is the name of the program currently residing in partition number *part numb.*

part numb is the partition number.

When the format ST,*part numb* is used, the status is printed as:

name

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).

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.

COMMENTS

The operator should give TM in response to the message printed when the RTE-III 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 TO L #*x* E #*y* S #*z*

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

This command is implemented by the disc resident program $$CMD. If $$CMD is not included in the operating system, the message NO SUCH PROG will be printed when the command is entered.

UP

Purpose:

The declare an I/O controller and all associated devices up (i.e., available for use by the RTE-III System).

Format:

UP,*eqt*

Where:

eqt is the EQT entry number of the I/O controller to be re-enabled.

COMMENTS

When the operator of the RTE-III System has set an I/O controller or device down for some reason, the operator should correct the situation before declaring the item 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 further requests (see LU, this section). Previous requests made to this device are switched to the new device. To prevent indefinite I/O suspension on a downed device, time-out is used. Refer to I/O Device Time-Out in Section V, and the TO command in this section.

The UP command places all downed devices (LU's) and the I/O controller (EQT) in the available state. Any I/O operations associated with downed devices are queued on the EQT for processing. If a device's problem has not been corrected, it will be reset down and an error message will be printed:

I/O NR E #*x* L #*y* S #*z*

SYSTEM COMMAND PROGRAM ($$CMD)

A foreground disc resident program, $$CMD, implements the EQ, LU, and TO system commands. If $$CMD does not exist in the system, the message NO SUCH PROG is printed when the commands are entered.

If the disc is down, the commands are unavailable. This situation can be avoided by making $$CMD memory resident during system generation.

ERROR MESSAGES

RTE-III rejects operator requests for various reasons. When a request is in error, RTE-III 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.
CMD IGNORED-NO MEM	Not enough system available memory exists for storing the program's command string. Re-enter the command (RU, ON, GO) or enter the inhibit (IH) form of the command.

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 NR L #*lu* E #*eqt* S #*sub*
I/O TO L #*lu* E #*eqt* S #*sub*

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 TO L #*lu* E #*eqt* S #*sub*
LU,*lu*,*eqt*

The error message says that the device at LU number *lu*, EQT number *eqt*, subchannel number *sub* 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). 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 normally printed on the system console.

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 line of code 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 CALL ABREG (IA,IB)
```

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

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

Table 3-1. RTE-III 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 Program Global	 4 15	Assigns a specific number of disc tracks for data storage.	 3-18
Disc Track Release Program Global	 5 16	Release assigned disc tracks	 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	 9 10 23 24	Schedules a program for execution. Immediate with wait. Immediate without wait. Queue with wait. Queue without wait.	 3-24
Time Request	11	Requests current real time.	3-26
String Passage	14	Retrieves program's command string or passes string to program's "Father."	3-27
Timed Execution Initial Offset Absolute Start	 12 12	 Schedules a program for execution after an initial offset. Schedules a program for execution at a specified time.	 3-29 3-31
Program Swapping Control	22	Allows a program to lock itself into core and notify system of core usage.	3-33
Resource Management		Allows cooperating programs a method of efficiently utilizing resources.	3-34
Logical Unit Lock Partition Status	– 25	Allows a program to exclusively dominate an I/O device. Provides information about a specified partition.	3-36 3-37

The Relocatable Library routine ABREG is used to pick up information left in the A- and B-Registers by EXEC. The ALGOL format is:

```
INTEGER IA,IB;
   :
PROCEDURE ABREG(A,B);
   INTEGER A,B;
   CODE;
    :
ABREG(IA,IB);
```

ASSEMBLY LANGUAGE FORMAT

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

```
EXT      EXEC     Used to link
 .                 program to RTE-III.
 .
 .
JSB      EXEC     Transfer control to
                  RTE-III.
DEF      *+n+1    Defines point of
                  return from RTE-III;
                  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 executive can be called through a CALL statement or as a function. The function is used when the user wants the A- and B-Registers returned in a variable.

For example, as a CALL:

```
CALL EXEC (ICODE, p2,...,pn)
```

Where

ICODE and *p2* through *pn* are either integer values or integer variables defined elsewhere in the program.

For example, as a function:

```
DIMENSION IREG(2)
EQUIVALENCE (REG,IREG,IA),(IREG(2),IB)
    :
    :
REG=EXEC (ICODE,p2...,pn)
```

The A-Register is returned in IA and the B-Register in IB.

ALGOL FORMAT

In ALGOL, the EXEC routine must be declared as an external CODE procedure before it is used. All formal parameters should be integers. Parameters that are not buffers and those that will not be modified by EXEC can be passed by value. When calling EXEC from ALGOL, pass a buffer by specifying the first element in an integer array (not the array name) as the actual parameter. For example, the following ALGOL program uses the EXEC routine with four parameters to perform a read:

```
HPAL,L,"MAIN"
   :
   :
INTEGER ICNWD;
INTEGER ARRAY IBUFR[20];
PROCEDURE EXEC(A,B,C,D);
   VALUE A,D; INTEGER A,B,C,D;
   CODE;
   :
   :
EXEC(1,ICNWD,IBUFR[1],20);
   :
   :
END$
```

The EXEC routine can be called with 1 to 9 parameters. However, ALGOL requires procedures to be called with a fixed number of parameters. If an ALGOL program calls EXEC with different numbers of parameters, unique procedure names must be declared within the main program and an external procedure must be written for each unique call. In the following example, a second EXEC call has been added to get the system time:

```
HPAL,L,"MAIN"
   :
   :
INTEGER ICNWD;
INTEGER ARRAY IBUFR[20], ITIME[5];
PROCEDURE EXEC(A,B,C,D);
    VALUE A,D; INTEGER A,B,C,D;
    CODE;
```

```
PROCEDURE EXECA(A,B)·
   VALUE A; INTEGER A,B;
   CODE;
    :
EXEC(1,ICNWD,IBUFR[1],20);
    :
EXECA(11,TIME[1]);
    :
END$
```

(External)

The following external procedure would be compiled separately:

```
HPAL,L,P,"EXECA"
PROCEDURE EXECA(A,B);
   VALUE A; INTEGER A,B;
BEGIN
PROCEDURE EXEC(A,B);
   INTEGER A,B;
   CODE;
EXEC(A,B);
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 then reschedules the program.

Assembly Language:

```
            EXT    EXEC
             :
            JSB    EXEC      Transfer control to RTE-III
            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.

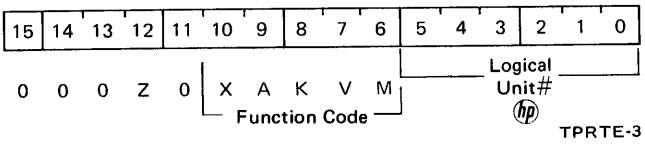


Figure 3-1. READ/WRITE (*conwd*) Format

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= 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 in words (only when the call is to be non-disc device). The Z-bit is passed to the driver.

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.

Modified to contain request code before entry into driver.

I/O AND SWAPPING

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

a. The buffer is not in the partition (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.

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 5 of Section IV, RTE-III 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-III
        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
IPRM2   DEC   q        Optional parameter
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-III 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.

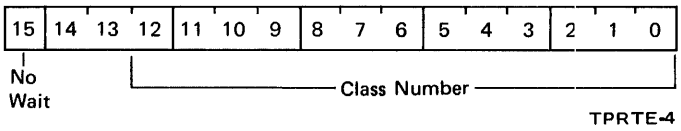


Figure 3-2. Class Number (ICLAS) Format

When the user's program issues a Class I/O call 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)=50122B
      INAME(2)=47507B
      INAME(3)=41000B
      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-III
            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.

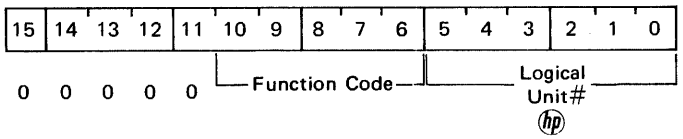


Figure 3-4 I/O Control (*conwd*) Format

Function Code (Octal)	Action
00	Clear device
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—leader skipped on next input request
10	Generate paper tape leader
11	List output line spacing
12	Write 3-inch gap (magnetic tape)

℗ *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).

The following functions are defined for the 2644 cartridge tape units (CTU). (Function codes 01, 02, 03, 04, 06, 13, and 14 have the same meaning for CTU as for magnetic tape.)

05 Rewind

10 Write end-of-file if not just previously written or not at load point

26 Write end-of-data

27 Locate file number IPRAM (less than 256)

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 as shown below.

IPRAM	Teleprinter	Line Printer
$+n$	space n lines	space n lines
$-n$	space n lines	top of form
0	no line feed	no line feed

*The dynamic status request (06) is unbuffered by RTIOC so that the caller receives the true status of any device. This causes the caller to wait for previous requests he (and lower priority programs) has made to be processed.

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-III
            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-III
                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)
ICODE = 21
IBUFL = n
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

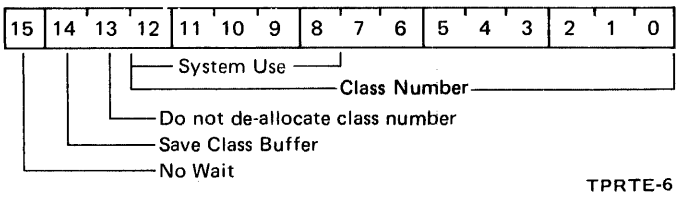


Figure 3-5. Class Word (ICLAS) Format

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-III
                DEF   RTN       Return address
                DEF   ICODE     Request code
                DEF   ICNWD     Control information
                DEF   ISTA1     Status word 1
                DEF   ISTA2     Status word 2 – optional
                DEF   ISTA3     Status word 3 – optional
RTN             return point    Continue execution (A and B are meaningless)
                 .
                 .
                 .
ICODE           DEC   13        Request code = 13
ICNWD           DEC   n         Logical unit number
ISTA1           NOP             Word 5 of EQT entry returned here
ISTA2           NOP             Word 4 of EQT entry returned here, optional
ISTA3           NOP             LU status returned here, optional
```

FORTRAN:

```
ICODE = 13              Request code
ICNWD = nn              nn is the logical unit number
CALL EXEC (ICODE,ICNWD,ISTA1,ISTA2,ISTA3)
```

COMMENTS

When this call is made the calling program is not suspended. Equipment Table (EQT) words 5 and 4 (optional) are returned in ISTA1 and ISTA2 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.

The status of the specified LU is returned in ISTA3. Bit 15 indicates whether the device (LU) is up (0) or down (1). Bits 4-0 give the subchannel associated with the device.

Table 3-3. I/O Status Word (ISTA1/ISTA2) 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)								

ISTA2

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).

ISTA1

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 subchannel 0 = interactive keyboard device (or system keyboard control devices)

subchannel 1,2 = HP mini-cartridge 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

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	–
2640 Terminal 2644 Terminal Cartridge Tape Unit DVR05	BF EOF	– TLP	CD EOT	– RE	– LCA	– CWP	TEN EOD	– CNE/DB
7210 Plotter DVR10	–	–	–	–	–	–	–	PD
2892 Card Reader DVR11	HE							RNR
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
7900 Moving Head Disc DVR31		NR	EOT	AE	FC	SC	DE	EE
7905 Moving Head Disc DVR32	PS	FS	HF	FC	SC	NR	DB	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 Track (protected)
AE = Address Error
EOT = End of Tape
NR = Not Ready
RY = Ready (0 = power on)
LCF = Last Character Flag
NE = No Error
DR = Disc Ready
HF = Hardware Fault
PS = Protect Switch Set
FS = Drive Format Switch is set
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
CD = Control-D Entered
BF = Buffer Flushed
CNI = Cartridge Not Inserted
EOD = End of Data
CWP = Cartridge Write Protected
LCA = Last Command Aborted
RE = Read Error
TLP = Tape at Load Point

DISC TRACK ALLOCATION

Purpose:

To request that RTE-III 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-III
        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-III stores starting track number here, or -1 if the tracks are
                       not available.
IDISC   NOP            RTE-III stores logical unit number here.
ISECT   NOP            RTE-III 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-III supplies only whole tracks within one disc. When writing or reading from the tracks (see READ/WRITE EXEC Call), RTE-III does not provide automatic track switching; the user program (when using this call) is completely responsible for file and track management. RTE-III 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-III 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-III
        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-III
        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 pro-
                                        gram 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 sus-
                                        pended for I/O, a system generated clear request is
                                        issued to the driver. 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) = 2Hcc     First two characters
            INAME(2) = 2Hcc     Second two
            INAME(3) = 1Hc      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),IREG(2)
EQUIVALENCE (IREG,REG),(IREG(2),IB)
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;
    .
    .
    .
EXEC (7);
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 7, Real-Time Programming, for information on segmented programs.)

Assembly Language:

```
                EXT   EXEC
                 .
                 .
                 .
                JSB   EXEC      Transfer control to RTE-III
                DEF   RTN       Return address
                DEF   ICODE     Request code
                DEF   INAME     Segment name
                DEF   IPRM1     }
                 .              }
                 .              } Up to five optional parameters
                 .              }
                DEF   IPRM5     }
RTN             return point    Control is transferred to the segment. (A = segment
                 .              ID segment address, B = as it was or parameter
                 .              address).
                 .
ICODE           DEC   8         Request code = 8
INAME           ASC   3,name    name is the segment name
```

FORTRAN:

```
DIMENSION NAME (3)
ICODE = 8
INAME (1) = 2Hcc        First two characters
INAME (2) = 2Hcc        Second two
INAME (3) = 1Hc         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).

The FORTRAN examples are HP FORTRAN IV. For HP FORTRAN, 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 and a buffer may be passed to the program.

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 schedule
                DEF   IPRM1  )
                 .           }  Up to five optional parameters
                 .           }
                DEF   IPRM5  )
                DEF   IPRM5
                DEF   IBUFR     Optional buffer address
                DEF   IBUFL     Optional buffer length

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  )

IBUFR  BSS n                    Optional buffer of n words
IBUFL  DEC n(or -2n)            Same n; words (+) or characters (-)
```

FORTRAN:

```
DIMENSION INAME(3),IBUFR(n)
IBUFL = n                       Set buffer length
ICODE = numb                    See ICODE above
INAME(1) = 2Hcc                 First two characters
INAME(2) = 2Hcc                 Second two
INAME(3) = 1Hc                  Last character
REG = EXEC(ICODE,INAME,IPRM1, . . . , IPRM5, IBUFR,IBUFL)
```

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 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 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)
```

If the "Father" includes the optional buffer in his scheduling call, the buffer is moved to system available memory and assigned to the "Son." The "Son" can recover the buffer by using the String Passage EXEC call. If there is not enough system available memory to hold the buffer, but there will be in the future, the "Father" is memory suspended. If there will never be enough memory available for the buffer, the "Father" will be aborted with a SC10 error. If the no abort bit (bit 15 in ICODE) is set, the program will not abort.

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.

For all schedule requests, the "Son" may pass back a buffer to the "Father" (see the String Passage EXEC 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-III
                    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., 1976).

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 6, Section V.

STRING PASSAGE

Purpose:

To retrieve the command string which scheduled the program or to pass a buffer back to the "Father" program.

Assembly Language:

```
        EXT     EXEC
         .
         .
         .
        JSB     EXEC    Transfer control to RTE
        DEF     RTN     Return address
        DEF     ICODE   Request code
        DEF     IRCOD   Retrieve/write code
        DEF     IBUFR   Buffer location
        DEF     IBUFL   Buffer length
RTN     return  point   Continue Execution (A = status, B = number
         .                 of words/characters)
         .
         .
ICODE   DEC     14      Request code
IRCOD   DEC     1 or 2  1 = retrieve parameter string
                        2 = write buffer to "Father"
IBUFR   BSS     n       Buffer of n words
IBUFL   DEC     n(or -2n) Same n; words (+) or characters (-)
```

FORTRAN:

```
      DIMENSION IBUFR(n)
      IBUFL = n
      ICODE = 14
      IRCOD = 1
      REG = EXEC(ICODE,IRCOD,IBUFR,IBUFL)
```

COMMENTS

The command string retrieved is exactly like the string used in scheduling the program via RU, ON, GO commands or EXEC 9, 10, 23 or 24. The block of system available memory used to store the command string (buffer) is released by this call or when the "Son" goes dormant. Any parsing of the returned string is left to the calling program. The RTE system library routine GETST can be used to recover the parameter string portion of the command string.

Upon return from a retrieve operation, the A-Register contains status information: 0 if the operation was successful or 1 if no string was found. The B-Register is a positive number giving the number of words (or characters) transmitted. If the string is longer than IBUFL, only IBUFL words are transmitted. If an odd number of characters are requested in a retrieve operation, the right half of the last word is undefined.

If the write parameter string option is used, the call returns any block of system available memory associated with the

"Father" and allocates a new block for the "Father" into which the string will be stored. If no memory is currently available, the "Son" is memory suspended. If there will never be enough memory, and bit 15 of ICODE is not set, the "Son" is aborted with an SC10 error. If there is no "Father," execution continues at the return point with the A-Register equal to 1. If the write parameter operation was successful, the A-Register is set to 0.

NOTE

Be careful when writing a buffer to a "Father" when the "Father" scheduled the "Son" without wait (EXEC 10 or 24). It is the user's responsibility to insure synchronization of the "Son's" write and the "Father's" read.

TIMED EXECUTION (Initial Offset)

Purpose:

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

Assembly Language:

```
            EXT   EXEC
             .
             .
             .
            JSB   EXEC      Transfer control to RTE-III
            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
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
```

FORTRAN:

```
            DIMENSION IPROG(3)      See IPROG above
            IPROG(1) = 2Hcc         First two characters
            IPROG(2) = 2Hcc         Second two
            IPROG(3) = 1Hc          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
            CALL EXEC (ICODE,IPROG,IRESL,MTPLE,-IOFST)
```

COMMENTS

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

RUN ONCE

After a time offset and the program to be scheduled is dormant, 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 and the program to be scheduled is dormant, 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-III places the specified program in the time list and returns to the calling program.

Assembly Language:

```
                EXT     EXEC
                 .
                 .
                 .
                JSB     EXEC        Transfer control to RTE-III
                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) = 2Hcc         First two characters
IPROG(2) = 2Hcc         Second two
IPROG(3) = 1Hc          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

After a time offset and the program to be scheduled is dormant, 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

After a time offset and the program to be scheduled is dormant, 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-III
        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.

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-1). 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-1). 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     Status 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.

15	14	5	4	3	2	1	0
Wait Option		Allocate Option			Set Option		
NO WAIT	NO ABORT	CLEAR	GLOBAL	LOCAL	CLEAR	GLOBAL	LOCAL

Reserved For System Use

TPRTE-7

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 globally.
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, x = 4 to set 'no abort', else x = 0
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.
                 .
                 .
                 .
NOLU            DEC     aa      Number of LU's to be locked/unlocked.
```

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.

UNLOCK

To unlock all owned LU's, the LUARY array is not used but still must be coded; the program will not abort.

Any LU's the program has locked will be unlocked when the program (1) does a standard terminate, (2) does a serial reusability terminate, or (3) aborts. Note that LU's will not be unlocked when the program does a "save resources terminate".

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.

PARTITION STATUS

Purpose:

To return information on the status of a particular partition. The status information provides the starting page number, the number of pages in the partition, whether the partition is reserved, and whether it is a real-time or background partition. If the partition contains a program, the index into the keyword table for the ID segment is returned.

Assembly Language:

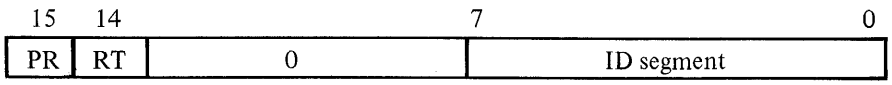
```
            EXT   EXEC
            .
            .
            .
            JSB   EXEC    Transfer control to RTE-III
            DEF   RTN     Return address
            DEF   ICODE   Request code
            DEF   IPART   Partition number
            DEF   IPAGE   Location where starting page is returned
            DEF   IPNUM   Location where number of pages is returned
            DEF   ISTAT   Location where partition status is returned
RTN         return point
            .
            .
            .
ICODE       DEC   25      Partition status request code = 25
IPART       DEC   n       Partition number
IPAGE       BSS   1       Starting page number returned here (0 if illegal partition number)
IPNUM       BSS   1       Number of pages returned here (-1 if illegal partition number)
ISTAT       BSS   1       Partition status returned here (see comments)
```

FORTRAN:

```
            ICODE=25
            IPART=n          where n is the partition number
            CALL EXEC (ICODE,IPART,IPAGE,IPNUM,ISTAT)
```

COMMENTS

The status word ISTAT contains status information as follows:



Where:

PR = 1 if the partition is reserved for programs requesting it.
= 0 if the partition is not reserved
RT = 1 for a real-time partition
= 0 for a background partition

If the partition contains a program, the index into the keyword table for the ID segment is returned in bits 0-7 of ISTAT; otherwise these bits are set to zero.

ERROR MESSAGES

When RTE-III 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-III 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
DM	Program tried to access a page not included in its logical memory (similar to MP).
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 = Not enough 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-III 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 (not 1, 2, 3, or 4).

SC07 = Prohibited core lock attempted.

SC10 = Not enough system available memory for buffer passage.

ERROR CODES FOR I/O CALLS

IO00 = Illegal class number. Outside table, not allocated, or bad security code.

IO01 = Not enough parameters.

IO02 = Illegal logical unit, or less than 5 parameters with X bit set.

IO03 = Not used.

IO04 = Illegal user buffer. Extends beyond FG/BG area or not enough system memory to buffer the request.

IO05 = Illegal disc track or sector.

IO06 = Reference to a protected track; or using LG tracks before assigning them (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.

LU03 = Not enough parameters furnished in the call. Logical unit reference less than one. Logical unit not locked to caller.

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						
IO00	Illegal Class Number 1. Outside Table. 2. Not allocated. 3. Bad Security Code.											
IO01	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								
IO02	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								
IO04	Illegal User Buffer. 1. Extends beyond FG/BG area. 2. Not enough system memory to buffer the request.	1										
IO05	Illegal Disc Track or Sector 1. Track number maximum. 2. Sector number 0 or maximum	1 2	1 2									
IO06	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									
IO07	Driver has rejected request and request is not buffered.	X	X	X								
IO08	Disc transfer implies track switch (LU2/3)	X	X									
IO09	Overflow of load-and-go area.		X									

PROGRAM SCHEDULE TIME 12	I/O STATUS 13	STRING PASSAGE 14	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					

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
IO10	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. 3. Not 4 parameters in String Passage Call.								1			
SC02	Illegal Parameter 1. Option word is missing or not 0, 1, 2, or 3. 2. Read/write word in String Passage Call is not 1 or 2.						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.											
SC10	Not enough system available memory for string passage.									X	X	

PROGRAM SCHEDULE TIME 12	I/O STATUS 13	STRING PASSAGE 14	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
									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	
													X	
2		3												
		2								1				
											2	2		
2											2	2		
										X				
		X									X	X		

SECTION IV
REAL-TIME PROGRAM PREPARATION

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

PART 1. RTE FORTRAN

The FORTRAN compilers accept 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 written to a device or stored in the LG tracks of the disc or both.

PART 2. 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 written to a device or stored in the LG tracks of the disc or both.

PART 3. RTE ASSEMBLER

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

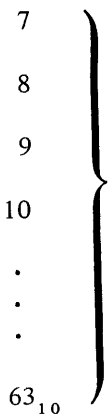
PART 4. RTE RELOCATING LOADER

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 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 Output Device
5	Standard Input Device
6	Standard List Device



Can be assigned to any devices by the user, for the defined range of logical units.

Note that LU8 is recommended as the magnetic tape device.

PART 5. RTE RELOCATABLE LIBRARY

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

PART 6. SEGMENTED PROGRAMS

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

PART 7. MULTIPLE TERMINAL OPERATION

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

PART 1
RTE FORTRAN

Regular FORTRAN and FORTRAN IV are segmented programs that execute in the background under control of RTE-III. The compilers consist of a main program and overlay segments, and reside in the protected area of the disc.

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 on an output device. The object programs can be loaded by the RTE Relocating Loader and executed by an ON operator request. When a FORTRAN program has been completely debugged, the RTE Relocating Loader can make it a permanent part of the RTE-III 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-III 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 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,FTN/FTN4

Purpose:

To schedule the FORTRAN compiler for operation.

Format:

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

or

RU, FTN4,*input, list, output, 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).

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

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

99 = The LG parameter (LG command required first). If present, the object program is stored in the LG 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 system will output a message to the operator:

I/O ET L *#lu* E *#eqt* S *#sub*

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-III 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 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 Memory-Resident
2 = Real-Time Disc-Resident
3 = Background Disc-Resident
4 = Not used
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.
The primary type may be expanded in some cases by adding 8, 16 or 24 to the number. These expanded types allow such features as access to real-time common by background programs and access to SSGA. See Appendix I for a list of expanded program types.

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 6 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.

ERR0 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 appropriate FORTRAN manual for the meaning of error codes.

PART 2
RTE ALGOL

The RTE ALGOL compiler is a segmented program that 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 Reference Manual (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*, *output*, *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).

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

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

99 = The LG parameter (LG command required first). If present, the object program is stored in the LG 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

A single ALGOL program can be made up of several source tapes. 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 system will output a message to the operator:

I/O ET L *#lu* E *#eqt* S *#sub*

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-III 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 LG 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, .RTOI, EXP, .ITOI, TAN.

type 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 it 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 = 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 *"name"* 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 console. The compiler then 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.

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 3
RTE Assembler

The RTE Assembler is a segmented program that consists of a main program and segments, and resides in the protected 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 written in binary records suitable for execution outside of RTE-III. ASMB can store relocatable code in the LG area of the disc for on-line execution, as well as writing it to an output device. The RTE Relocating Loader accepts Assembler Language relocatable object programs.

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 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*, *output*, *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).

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

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

99 = LG 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 system will output a message to the operator:

I/O ET L *#lu* E *#eqt* S *#sub*

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-III 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 console 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 console signals:

$END ASMB NPRG

The current assembly aborts.

RTE-III generates two messages when ASMB attempts to write on the LG tracks (ASMB 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 have overflowed. The operator must define more LG tracks with LG and start the assembly 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.
The primary type may be expanded in some cases by adding 8, 16, or 24 to the number. These expanded types allow such features as access to real-time common by background programs and access to SSGA. See Appendix I for a list of expanded program types.

pri	is the priority (1 to 32767, set to 99 if not given).	
res	is the resolution code	(Time values, set to 0 if not given. See Section II, IT, for meaning)
mult	is the execution multiple.	
hr	is hours.	
min	is minutes.	
sec	is seconds.	
msec	is tens of milliseconds.	
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.

PART 4
RTE RELOCATING LOADER

The RTE-III On-line Relocating Loader, LOADR, accepts relocatable code from an input device such as paper tape, magnetic tape, and so forth, or from LG tracks (see LG operator request, Section II) filled by the RTE Assembler, RTE FORTRAN, or RTE ALGOL. LOADR provides for linking the relocatable program file produced by the Assembler or the compilers together with one or more library files.

The resultant program may be relocated (loaded) as a background program; or it may be relocated as a background program with the DEBUG library routine appended to it; or it may be relocated as a real-time disc-resident program for subsequent action. In addition, LOADR may be used to list program names and blank ID segments, purge permanent programs from the system, add permanent programs to the system, or replace permanent programs in the system.

The RTE-III Relocating Loader has the following features:

- Can be operated under control of the File Manager in batch mode.
- LOADR is swappable and can be operated in either real-time or background disc-resident areas.
- Allows programs which declare COMMON to reference either a system or reverse common area (shared with other programs) or a local common area (not shared with other programs).
- Can relocate referenced library routines which are on 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 loader can be used to purge a permanent program; the OF,*name*,8 command will not remove a permanent program from the system).
- Allows a program to be temporarily loaded into either the real-time or background area.
- Allows a program to reference absolute and code replacement type ENT records (see Section VI).
- Uses system area disc tracks that have been vacated by deleted programs.
- Uses a *short* ID segment (when available; see Appendix A) when loading a background program segment. In addition, the loader does not restrict the arrangement of subroutines following segments (e.g., if a subroutine is shared by two segments, only one copy of it is necessary on the LG tracks).

Options are available as parameters to the ON,LOADR statement which permit you to specify:

1) the logical unit number of the input device.
2) the logical unit number of the list device.
3) an operation code which includes a Subsystem Global Area (SSGA) flag together with common type and loader operation indicators.
4) a program format code which includes program memory size requirements, partition assignment, and a program structure indicator.
5) listing characteristics.

A detailed description of the ON,LOADR statement is given under Loader Operation in this section.

At load time, it is not necessary to know the *real* address of the partition in which the program will run because each partition appears to be within the first 32K words of memory. The location at which a program area appears to begin is a *logical* address. The program is relocated with respect to this logical address. Logical memory address space configurations are illustrated in Section VI, Figure 6-2.

You have more address space available for your program if it does not declare COMMON. If COMMON is not declared, the program is relocated (loaded) to begin after the system resident library. If COMMON is declared, the common area begins after the system resident library and is followed by the program.

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: $END, or /LOADR ABORTED, the LG track area is cleared. If the loader terminates in some other manner, the LG track area is not cleared.

PROGRAM RELOCATION

During loading, programs are relocated to start at the beginning of the disc-resident program area of logical memory. If COMMON is declared, the program will be preceded by the common area. The logical address of the program location is always at a page boundary and the first two words of the program location are allocated for saving the contents of the X- and Y-registers whenever the program is suspended.

Once relocated, the program is linked to external references such as EXEC, the resident library, or the relocatable library. Any segments will overlay the memory area immediately following the main program and its subroutines.

LOADR stores the absolute version of the program, its subroutines, and linkages on a disc track or a group of contiguous disc tracks; it then assigns the disc tracks to the system (that is, they are not available as scratch or data tracks to programs). LOADR then builds or updates the ID segment assigned to the program. The program together with its subroutines and its largest segment may be as large as the largest partition of the same type. If a program is assigned to a partition, it must not be larger than the partition or an error, L17, results (see LOADR Error Messages). Common may be allocated in one of several areas according to the needs of the programmer (see the optional parameter list for the RU,LOADR request).

PROGRAM DISPOSITION

Regardless of the program type recorded in the program's NAM record, the third parameter *(opcode)* of the RU,LOADR request determines the program disposition. This parameter's default (zero) results in a background temporary program. Other parameter values result in a background temporary program with the DEBUG library subroutine appended, a real-time temporary program, a real-time or background permanent program addition, or a real-time or background permanent program replacement.

ON-LINE MODIFICATION

Using the loader, the operator can permanently modify the set of disc-resident user programs in a configured RTE-III 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 request cannot delete program segments that were permanently added on-line or stored during generation.

When the system is generated, RTGEN, the system generator, stores disc-resident programs on the disc in an absolute, packed format. Each main program is identified and located by a 28-word ID segment. The ID segments are stored in the ID segment area of the system on the disc and brought into main memory when the system is started up. For disc-resident programs, the program's disc location as well as its main memory and base page addresses are kept in the ID segment. When a main program and segments are loaded, the segments are identified and located by a 9-word short ID segment. See Appendix A for ID segment format.

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 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 ID segment 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.

Memory-resident programs cannot be replaced in the system because the length of the program and linkage area is not kept in the ID segment for these programs, nor can programs be added.

If a user supplied routine is to be referenced by a program, and a memory-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.

When performing an on-line modification, the disc hardware protect must be physically disabled prior to the loading (and then enabled afterwards) unless the protection is always kept disabled. RTE provides additonal software protection for any tracks containing system programs or user programs.

LIMITATIONS

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

a. System or reverse common is requested but the program's common length exceeds that of the common area.

Local common is requested and COMMON is not declared by the first relocatable module encountered by the loader, even though the module is a dummy module which contains no executable code.

b. The base page linkages exceed the corresponding linkage area for disc-resident programs established by the system during generation.

c. The length of the absolute program unit exceeds the area available.

d. Disc space is not available to store the program.

e. A blank ID segment is not available for adding a program. (A program previously loaded can be deleted to create a blank ID segment.)

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 will not necessarily result in the main and segments being stored on contiguous tracks.

When replacing segmented programs that were incorporated into the system at generation time, the operator must replace every segment with a new segment having the same name, or remove the original segment permanently from the system. Additonal 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, assigned partition, 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.

PROGRAM REPLACEMENT

In a replacement, the new program uses the ID segment of the old program (both programs must have the same name). The new program is generated onto temporary tracks, and then, if it can fit in the old area, or within another area gained by deleting a program incorporated during generation, it is transferred to that area. If not, the temporary tracks attain system track status and if a blank ID segment is available, the old tracks are assigned to it for later use by another program. If a blank ID segment is not available, the old tracks are lost.

PROGRAM DELETION

A temporary program is deleted from the system with the OF,*name*,8 command. A permanent program (i.e., a program loaded during generation, or on-line with the loader as a permanent addition or replacement load) is deleted with the loader. When using the loader to delete a permanent program, the *opcode* parameter is set to 4. This blanks the program's ID segment making it available for loading another program. The tracks containing the program are released, unless they are system tracks. If the program had been saved through the File Manager on FMP tracks, those tracks are not released to the system but remain as FMP tracks.

COMMON ALLOCATIONS

There are three options you can specify when allocating common area for a program.

SYSTEM COMMON. This implies a background program with common in the background system common area, or a real-time program with common in the real-time common area.

LOCAL COMMON. Common area for a background program is established at the beginning of the background program's area, or for a real-time program, at the beginning of the real-time program's area. The common area will be swapped together with the program.

REVERSE COMMON. This implies a background program with its common in the real-time common area. Conversely, a real-time program can reference and use the background system common area.

LOADER OPERATION

The operator schedules the loader for execution with the ON operator command.

RU,LOADR

Purpose:

To relocate a program so the program can be scheduled by an ON or RU operator request.

Format:

RU,LOADR,*input,list,opcode,fmt,listing*

Where:

input = Parameter 1, – logical unit number of input device. If set to 99, the LG tracks are used, but 99 does not terminate the parameter list (default = 5).

list = Parameter 2 – logical unit number of list device (default = 6).

opcode = Parameter 3 – SSGA Flag/Common Type/Operation Code: A three digit code (decimal) indicating Subsystem Global Area Flag, Type of Common, and Loader Operation to perform, as follows:

abc

where,

Subsystem Global Area Flag	*a* = 0	No SSGA (default)
	= 1	Use SSGA
Type of Common	*b* = 0	No Common or use local Common if any is declared (default)
	= 1	System Common
	= 3	Reverse Common
Loader Operation	*c* = 0	Background Temorary (default)
	= 1	Background (DEBUG appended)
	= 2	On-line Edit
	= 3	List
	= 4	Purge
	= 5	Real-time Temporary

= 6 Real-time Replacement
= 7 Real-time Addition
= 8 Background Replacement
= 9 Background Addition

fmt = Parameter 4 – Size/Partition/Structure Control: A five digit code (decimal) indicating memory size and override requirements, partition assignment, and program structure as follows:

xxyyz

where,

Size	*xx* =	00	Use program size (default)
		01-32	Required size in pages
Partition	*yy* =	00	None assigned (default)
	=	01-64	Assigned partition number
Structure	*z* =	0	Main program (default)
	=	1	Main program plus segments

listing = Parameter 5 – Listing parameter (only when *input* = 99). A one digit code

Where:

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

If a track allocation cannot be made, the message WAITING FOR DISC SPACE is printed. The loader repeats the disc request and is suspended until space becomes available.

Following the relocation of a program which has its external references satisfied, the loader terminates with the following messages:

/LOADR: *name* READY
/LOADER: $END

Where *name* is the main program name. The loader terminates and the program is ready to run.

opcode **PARAMETER (PARAMETER 3)**

This parameter is specified as a decimal value of three digits. The first (left) digit determines whether or not the program uses the Subsystem Global Area (SSGA). The second digit determines whether or not common is to be used and, if common is used, the type of common. The third digit determines the loader operation to be performed. Setting the loader operation value to 1 causes the DEBUG utility routine to be appended to each main program and segment. The loader sets the primary entry point of each main program and segment to DEBUG, rather than to the user program. When the program is run, DEBUG takes control of program execution and requests instructions from the keyboard. (See RTE DEBUG LIBRARY SUBROUTINE for legal commands.)

Setting the loader operation value to 2 initiates the loader for an on-line edit which requires additional information through the GO,LOADR request. See Loader Operation (On-line Edit) for addition information.

Setting the loader operation value to 3 causes a listing of all program names and blank ID segments to be printed. For each ID segment in the system the following format is used:

name, type, priority, partition

name is the program name

type is the program type:

1 – Real-time Memory-resident.
2 – Real-time Disc-resident.
3 – Background Disc-resident.
4 – Background Memory-resident.
5 – Background Segment.

priority is the program priority, from 1 to 32767.

partition is the partition number, from 1 to 64. This field appears only if the program is assigned to a specific partition.

Each blank ID segment available for use by the loader is noted by the line:

<LONG BLANK ID>

– or –

<SHORT BLANK ID>

The loader terminates after the list is complete.

Setting the loader operation value to 4 initiates the purge of 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 LOADR) enter /A.

Setting opcode = 5-9 allows the loader to run in Batch mode without operator intervention.

fmt PARAMETER (PARAMETER 4)

The *fmt* parameter is specified as a decimal value of five digits. The first two (left) digits determine whether or not memory space larger than the program size is required. For example, programs such as EDITR, ASMB, and LOADR require extra memory space to store symbol tables, intermediate buffers, and so forth. If space larger than the program size is required, the total space required is specified in pages of memory; if program size is sufficient, set these digits to zero.

The next two digits determine whether or not the program is to be assigned to a specific partition. A program may be assigned to any partition of sufficient size regardless of partition class (i.e., real-time or background). If a program is to be assigned to a partition, specify the partition number desired, otherwise, set these digits to zero. If zero, the program executes in the smallest available partition possible when it is scheduled.

The LG track area must contain the main program and its subroutines followed by a segment and its subroutine. Where the same subroutine is required by both the main program and segment, but is not in the library, it need only appear in the main program. Subroutines required by more than one segment, but not by the main, can appear with each segment in the LG track area for greater speed in loading. Or, only once copy of the subroutine may be placed in the LG track area and the area is scanned as a library (loading time is somewhat longer).

NOTE

If the loader suspends as a result of undefined external references in any segment, you may move additional subroutines to the end of the LG track area and scan the area as a library, or have the loader scan the input device for required subroutines. In Batch mode, the loader aborts.

If there are undefined external references in the main program, LOADR printes the message MAIN followed by the message UNDEFINED EXTS and then suspends. It is not possible to satisfy these undefined external references using the loader. The only acceptable request is GO,LOADR,4 to continue loading without satisfying external references, or GO,LOADR,98 to list the undefined external references.

If the last digit is 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 segment).

The loader scans the relocatable programs and subroutines as it reads them, keeping track of any external references. If input is initially from the disc as specified by 99 in the ON statement, the loader immediately scans the library for entry points. If input is from paper tape and DVR00 has been configured to logically disable (down) the tape reader on end-of-tape (EOT condition), the loader suspends operation with the message:

I/O ET L *#lu* E *#eqt* S *#sub*
/LOADER: LOAD

LU *#lu* is unavailable (down, see DN, Section II) until the operator declares it up:

UP,*eqt*

If an EOT condition does not cause the tape reader to be set down, the RTE system does not send any messages but does suspend the loader.

LOADER RESCHEDULING

After the loader has been suspended, the operator can reschedule it with the GO operator command.

GO,LOADR (PROGRAM RELOCATION)

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 track area. LG track area has 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. Ignore undefined external references.
= 98 – List undefined externals.
= 99 – Load from LG track area 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 (forced relocation).
= 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 LG track area upon a successful load.

input option **PARAMETER**

- Once you have force loaded from the LG track area, subsequent use of the 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 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 forced relocation.

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 a forced relocation. 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 points to external references, if there are still undefined external references, the loader prints this message:

UNDEFINED EXTS

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

To load additional 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, *name* 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 (override) 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.
= 3 for a background disc-resident program.

priority = priority (optional) from 0 to 32767 (a 0 means use the priority value in the NAM record of the program or, if that priority is 0, uses 32767).

COMMENTS

Any errors encountered while attempting to reschedule the loader cause the error message L10 to be printed (see LOADR ERROR MESSAGES). 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 ET L *#lu* E *#eqt* S *#sub*
/LOADER: LOAD

LU *#lu* is unavailable (down; 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 checked for logical errors during execution. Programs that expect starting parameters or that call RMPAR (see Section III, Program Suspend Exec Call) cannot use DEBUG because DEBUG uses these parameters.

The DEBUG routine was developed for 2100 series computers. It should not be used with instructions labeled as "21MX Only" in the RTE Assembler Reference Manual (92060-90005).

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 program name.

The primary entry point of the 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

You can then enter any legal debug operation. Illegal requests are ignored and a message is printed.

ENTRY ERROR

The following commands describe DEBUG operations.

B,*n*	Instruction breakpoint at octal address *n* (Note: if *n* = JSB EXEC, a memory protect violation occurs)
D,A,n_1[,n_2]	ASCII dump of octal main memory address n_1 or from n_1 through n_2
D,B,n_1[,n_2]	Binary dump of octal main memory address n_1 or from n_1 through n_2
M,*n*	Sets absolute base of relocatable program unit at octal address *n*
R[,*n*]	Execute user program starting at octal address *n* or 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,*n*,*d*	Set octal value *d* in octal address *n*
S,*n*,d_1,d_2, . . . ,d_n	Set octal values d_1 through d_n in successive memory locations beginning at octal address *n*
W,A,*d*	Set A-register to octal value *d*
W,B,*d*	Set B-register to octal value *d*
W,E,*d*	Set E-register to octal value *d* (0 = off; non-zero = on)
W,O,*d*	Set Overflow to octal value *d* (0 = off; non-zero = on)
X,*n*	Clear breakpoint at octal address *n*
A	Abort DEBUG operation.

LOADR ERROR MESSAGES

Messages are printed in this format:

/LOADR: *message*

WARNING (W) MESSAGE

W17 – Number of pages required by the program exceeds the partition size. The loader cannot find a partition large enough for the program. It can be relocated successfully but cannot be executed. You can generate a new system containing a partition large enough for the program or you can revise the program.

L ERROR MESSAGES

L01 – Checksum error

L02 – Illegal record

These errors are recoverable (except in Batch mode). The record in error can be reread by repositioning the tape to the beginning of the record 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 the LG track area is specified, but no program exists in that area.)

L09 – Record out of sequence.

L10 – Operator request parameter error. GO requests may be retyped; ON requests may not.

L11 – Operator attempted to replace or purge a memory resident program.

L12 – LG track area used without presetting (*input option* = 2 IN 'GO'). *input option* was not specified as 99 previously.

L13 – LG track area has been illegally reset (i.e., overwritten). Program addition on area not allowed if it has already been specified for program input, or area was once used for force loading with *input option* = 99 and it is again being used with *input option* = 99 (must = 2).

L14 – Assembler or compiler 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 an indirect forward reference.

L16 – Illegal partition number. Value must be in the range from 1 through 64.

L17 – Number of pages required by the program exceeds assigned partition size. A specific partition has been assigned for this program. This program requires more pages than are available in the partition.

L18 – Total number of pages required exceeds 32. The sum of required pages must be in the range from 1 through 32.

ADDITIONAL MESSAGES

NO BLANK ID SEGMENTS

This message is printed when no available (i.e., blank) ID segment is found. The loader calls for program suspension. The operator may then delete a program from the system (OF,*name*,8 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 to scan the library again.

PART 5
RTE Relocatable Library

INTRODUCTION

DOS/RTE Relocatable Library Reference Manual (24998-90001) describes the subroutines contained in the following relocatable libraries:

Library Mnemonic	Library Name
RLIB.N	DOS/RTE Relocatable Library
FF4.N	FORTRAN IV Formatter
FF.N	FORTRAN Formatter

In addition to the above libraries there is a system library which contains routines unique to the RTE-III 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-III through entry point $LIBR to save the data. At the end of execution, the re-entrant routine calls RTE-III 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-III to save temporary
                      data.
        DEF   TDB     Address of temporary data.
        .
        .             Program instructions
        .
EXIT    JSB   $LIBX   Call RTE-III to restore data.
        DEF   TDB
        DEC   m       m is for routines with two return
                      points in the calling program; 0
                      specifies the error-point 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
 .
 .                     Temporary data (n words).
 .
Tn
```

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.

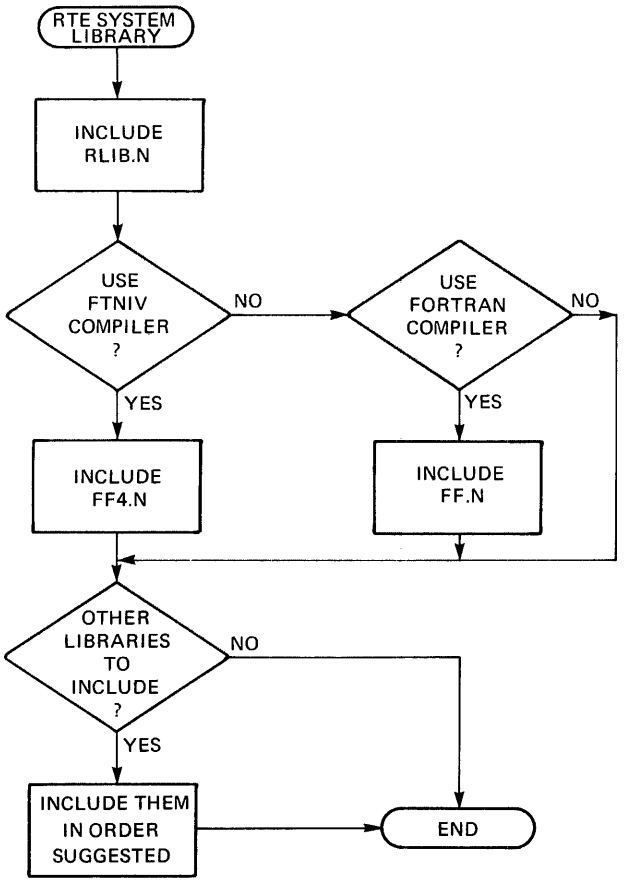


Figure 4-1. RTE Library Selection

FORMAT OF PRIVILEGED ROUTINE

```
      NAM xxxx,6
      EXT $LIBR,$LIBX

ENTRY NOP          Entry points to the routine.
      JSB $LIBR    Call RTE-III to disable the interrupt
                   system and memory protect fence.
      NOP          Denotes privileged format.
       .
       .
       .
EXIT  JSB $LIBX    Call RTE-III 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) 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 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 internal design or I/O operations. A copy of the utility routine is appended to every program that calls for it. The library subroutine FRMTR (FF .n) or FMTIO (F4D.n) which carries out FORTRAN I/O operations, and the PAUSE subroutine are examples of utility routines.

When the RTE-III System is generated, all library subroutines not included in the resident library are loaded immediately following each user program requiring them during program relocation.

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 described in the READ/WRITE EXEC call in Section III of this manual. Note that REIO can only be used with READ/WRITE calls and that the optional parameters available in those calls are not allowed in the REIO call. 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. 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 (IBUF1,ICCNT,IBUF2)
```

Where

IBUF1 is an array containing the string to be parsed
ICCNT is the character count
IBUF2 is a 33 word array that will contain the result of the parse.

BINARY TO ASCII CONVERSION SUBROUTINES

The $CVT3 subroutine is used to convert an integer binary number of ASCII. Note that the calling program must be privileged when using the following 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. (For CNUMD, the number must not be negative.)

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.

If the request is LU, EQ, or TO (starting in first column), any resulting message will be written to the device specified by LU. If LU is not given, the message will be sent to the system console. No message is returned to the caller for these commands.

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. (It is not needed for ALGOL.)

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

RECOVER PARAMETER STRING

The routine GETST recovers the parameter string from a program's command string storage area. The parameter string is defined as all the characters following the second comma in the command string (third comma if the first parameter is NO). The Assembly Language calling sequence is:

```
        EXT GETST
         .
         .
         .
        JSB GETST       Call subroutine
        DEF RTN         Return address
        DEF IBUFR       Buffer Location
        DEF IBUFL       Buffer length
        DEF ILOG        Transmission Log
RTN     return point    Continue execution
         .
         .
         .
IBUFR   BSS n           Buffer of n words
IBUFL   DEC n(or -2n)   Same n; words (+)
                        or characters (-)
ILOG    NOP             Error information
```

Upon return, ILOG contains a positive integer giving the number of words (or characters) transmitted. The A- and B-Registers may be modified by GETST. Note that if RMPAR is used, it must be called before GETST.

When an odd number of characters is specified, an extra space is transmitted in the right half of the last word.

PART 6
Segmented Programs

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 partition (after base page), 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 .

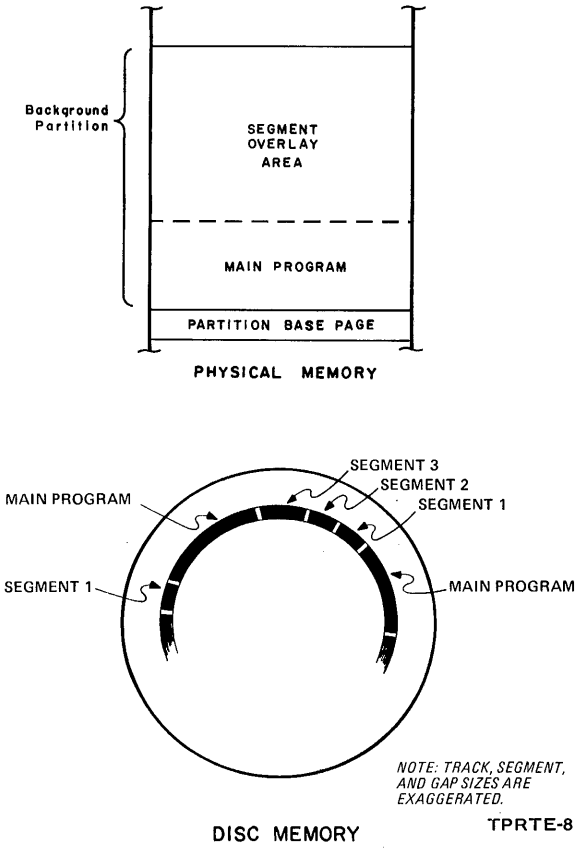


Figure 4-2. Segmented Programs

RTE ALGOL SEGMENTATION

ALGOL programs can be segmented if certain conventions are followed. The main program must be primary 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 primary 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. Segments can call

any subroutine attached to the main program. Communication between the main program and segments may be through COMMON.

RTE ASSEMBLER SEGMENTATION

The main program must be primary 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.

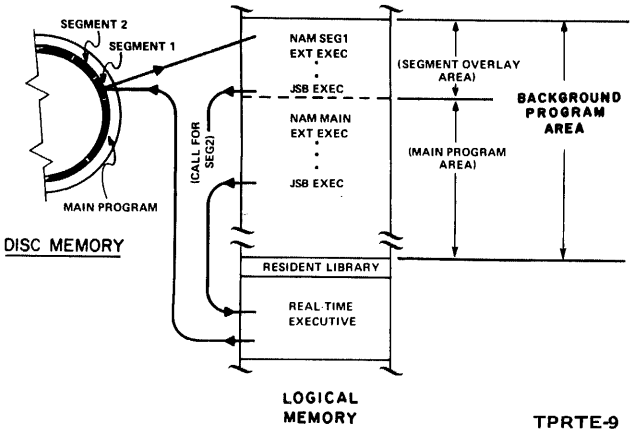


Figure 4-3. Main Calling Segment

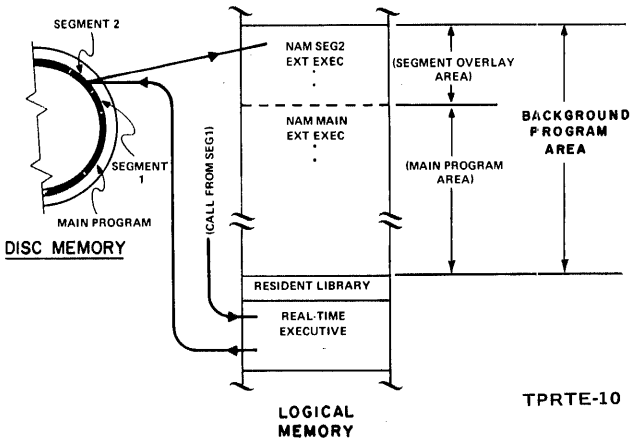


Figure 4-4. Segment Calling Segment

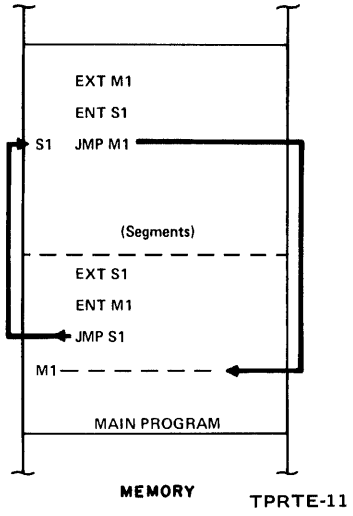


Figure 4-5. Main-to-Segment Jumps

PART 7
Multiple Terminal Operation

Through Class Input/Output, which consists of initiating an I/O request without wait, the RTE-III 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 immediately 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-III 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-III 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

In the Real-Time Executive System, centralized control and logical referencing of I/O operations effect simple, device-independent programming. Each I/O device is interfaced to the computer through an I/O controller associated with one or more I/O channels which are linked by hardware to corresponding memory locations for interrupt processing. By means of several user-defined I/O tables, self-contained multi-device drivers, and program EXEC calls, RTE-III 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 controller's I/O channels, driver, DMA, buffering and time-out specifications. A Device Reference Table assigns one or more logical unit numbers to each device and points each device to the appropriate Equipment Table entry, 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-III can call a driver (which is responsible for initiating and continuing operations on all devices' controllers 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 subsystems may require additional parameters.

THE EQUIPMENT TABLE

The Equipment Table (EQT) has an entry for each I/O controller recognized by RTE-III (these entries are established by the user when the 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	I/O Request 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 (EQT extension size)															
13	Temporary Storage for Driver (EQT extension starting address															
14	Device Time-Out Reset Value															
15	Device Time-Out Clock															

*Modified by RTE at each I/O initialization.

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 the I/O controller (lower number if a multi-board interface).
AV	=	I/O controller availability indicator:
0	=	available for use.
1	=	disabled (down).
2	=	busy (currently in operation).
3	=	waiting for an available DCPC 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 devices on this controller. When this octal number is linked with "DV" 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.
05 sub 0	=	console (or system Keyboard control device).
05 sub 1	=	mini cartridge.
05 sub 2	=	devices
10 to 17	=	unit record device.
10	=	plotter.
11	=	card reader.
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).

When RTE-III initiates or continues an I/O operation, it places the addresses of the EQT entry for the device's controller 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 by the EQT (I/O controller) and the subchannels (if applicable) and also define the physical devices' (LU) status. These numbers are maintained in a two word Device Reference Table (DRT), which is created during system generation, and can be modified by the LU operator request (see Figure 5-1).

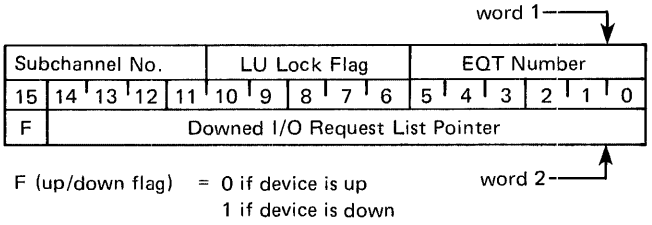


Figure 5-1. Device Reference Table

DRT word one contains the EQT entry number of the device assigned to the logical unit, and the subchannel number within the EQT entry. The second DRT word contains the logical unit's status (up or down) and a pointer to any downed I/O requests. If the pointer is less than 64, it is the LU number off of which the downed I/O requests are queued. If several LU's point to the same device, the requests are queued off the lowest LU number (the major LU). If the pointer is greater than 64, it points to the device's downed I/O request list. There are separate tables for words one and two, with the word two table located in memory immediately following the word one table. The starting address and length of the word one table are recorded in the base page. The functions of logical units 0 through 6 are predefined in the RTE-III System as:

0 – bit bucket (null device)
1 – system console
2 – system disc
3 – auxiliary disc
4 – standard output 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-III 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 controller 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-III prints this message:

ILL INT *xx*

where *xx* is the octal I/O channel number. RTE-III 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 I/O controller on the channel.

The interrupt locations in memory 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-III 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's controller 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 request list corresponding to the referenced I/O controller. The parameters from the request are set in the temporary storage area of the Equipment Table.

If the device's controller is available (i.e., no prior requests were stacked), the "initiation" section of the associated driver is called. The initiation section initializes the device's controller and starts the data transfer or control function. On return from the initiation section, or if the device's controller 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's controller (EQT) or the device (LU) 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 LU or EQT 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's controller 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 an I/O stacked request. If there are no stacked requests, IOC exits to the dispatching module (DISP); 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, it 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 I/O controller. The status of bits 14 and 15 will indicate whether the I/O controller 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 I/O controller 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 controller is set "down." The system then sets the controller "up," resets the EQT and enters the driver at I.*nn*.

In other words, if the controller was reading or writing data when the power failure occurred and the driver is designed to handle power fail, when power resumes the controller driver will do the power fail recovery. If the controller was busy when power failure occurred and the controller driver is not written to handle power failure, the routine attempts to restart the I/O operation.

d. If the controller or 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 controller or device was down when power failed, when power resumes the system "ups"

the controller and device and attempts to start the operation, if any, in the controller I/O request list.

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-III, 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 DCPC channel, if required. It should be noted that RTE-III operation makes it manadatory that a synchronous device driver use a DCPC or privileged interrupt channel for data transfer.

Many of the I/O drivers and subroutines are documented in the RTE Driver and Device Subroutine Library manuals; HP Part Nos. 92200-93005 and 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 distribution package. 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 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-III 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's controller, 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 channel (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-III 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 and 1 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 or controller 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's controller.

d. Initializes DMA, if appropriate.

e. Initializes software flags and activates the device's controller. All variable information pertinent to the transmission must be saved in the EQT entry

because the driver may be called for another controller before the first operation is complete.

f. Optionally set the device's controller 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.

DMA INITIALIZATION – A driver can obtain a DCPC 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 DCPC 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 DCPC 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.)

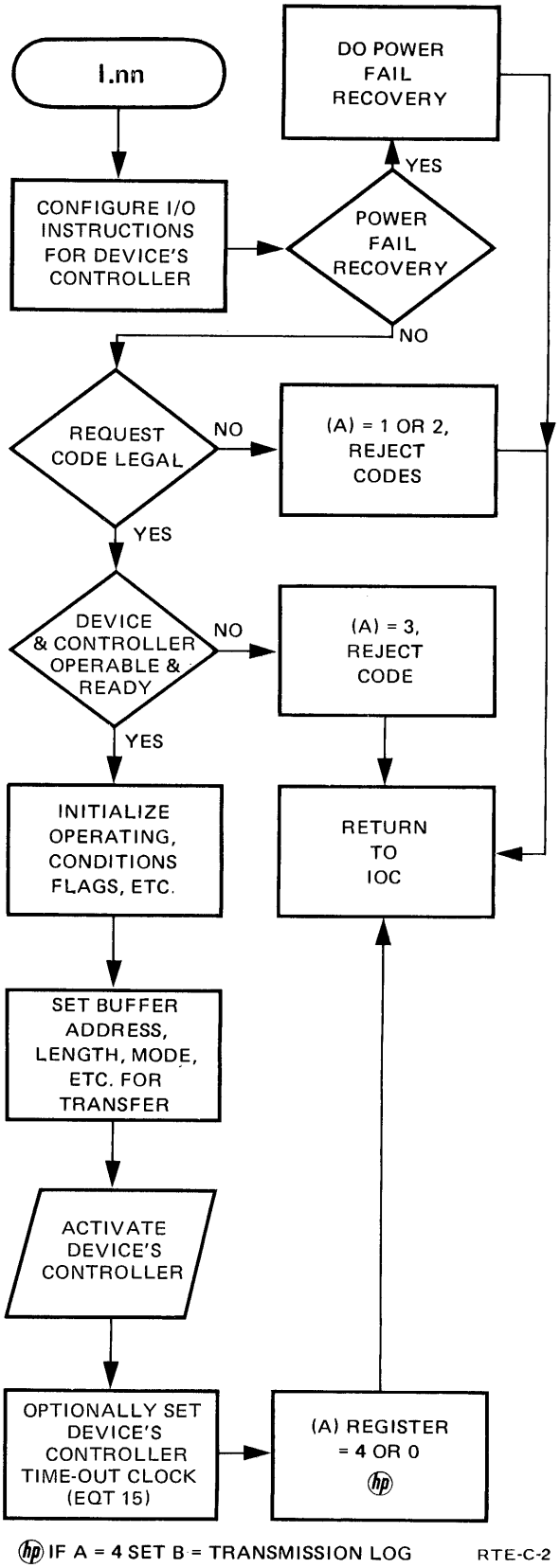


Figure 5-2. I/O Driver Initiation Section

The following code demonstrates these principles:

```
        CLF  0              Disable interrupts.
        STC  DMA,C          Initiate DCPC channel.
        CLA                 Bypass this section if DUM-
        CPA  DUMMY          MY = 0 and special processing
        JMP  X              is not needed (go to X).
        CLC  DMA
        LDB  INTBA          Clear DMA control. Set B =
        LDA  CHAN           address of the appropriate en-
        CPA  = D7           try in the interrupt table.
        INB
        LDA  B,I
        IOR  = B100000      Set bit 15 of the entry equal
        STA  B,I            to 1 and return to the inter-
        STF  0              rupt table. Enable interrupt
                            system.
X EQU *                     Continue processing.
```

There may be times when a driver will only occasionally need DMA, and thus not want to always tie up a DCPC channel while it is operating. This may be done in one of two ways: (Note that in example No. 1, the DCPC channel is always assigned before the driver is entered. In example No. 2, the DCPC 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 DCPC 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 DCPC 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 DCPC 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 DCPC 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 DCPC 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-III calls the completion section of the driver whenever an interrupt is recognized on an I/O controller 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.

a Checks whether word 1 (controller I/O request 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 controller). 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 intructions in the completion section to reference the interrupting controller, and then proceeds to "b."

b. If both DMA and the device controller completion interrupts are expected and the device controller 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 controller 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 or controller malfunction or not ready.
2 – end-of-tape (information).
3 – transmission parity error.
4 – device time-out.

f. Clears the device controller and DMA control, if end-of-operation, or sets the device controller 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 controller 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 CONTROLLER TIME-OUT

Each I/O controller 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's controller fails to return a flag (possible hardware malfunction or improper program encoding). Without the controller 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's controller. 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 controller function as a controller time-out clock. EQT 15 is the actual working clock, and before each I/O transfer is initiated, is set to a value *m*, where *m*, is a negative number of 10 ms time intervals. If the controller does not interrupt within the required time interval, it is to be considered as having "timed out." The EQT 15 clock word for each controller 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=)

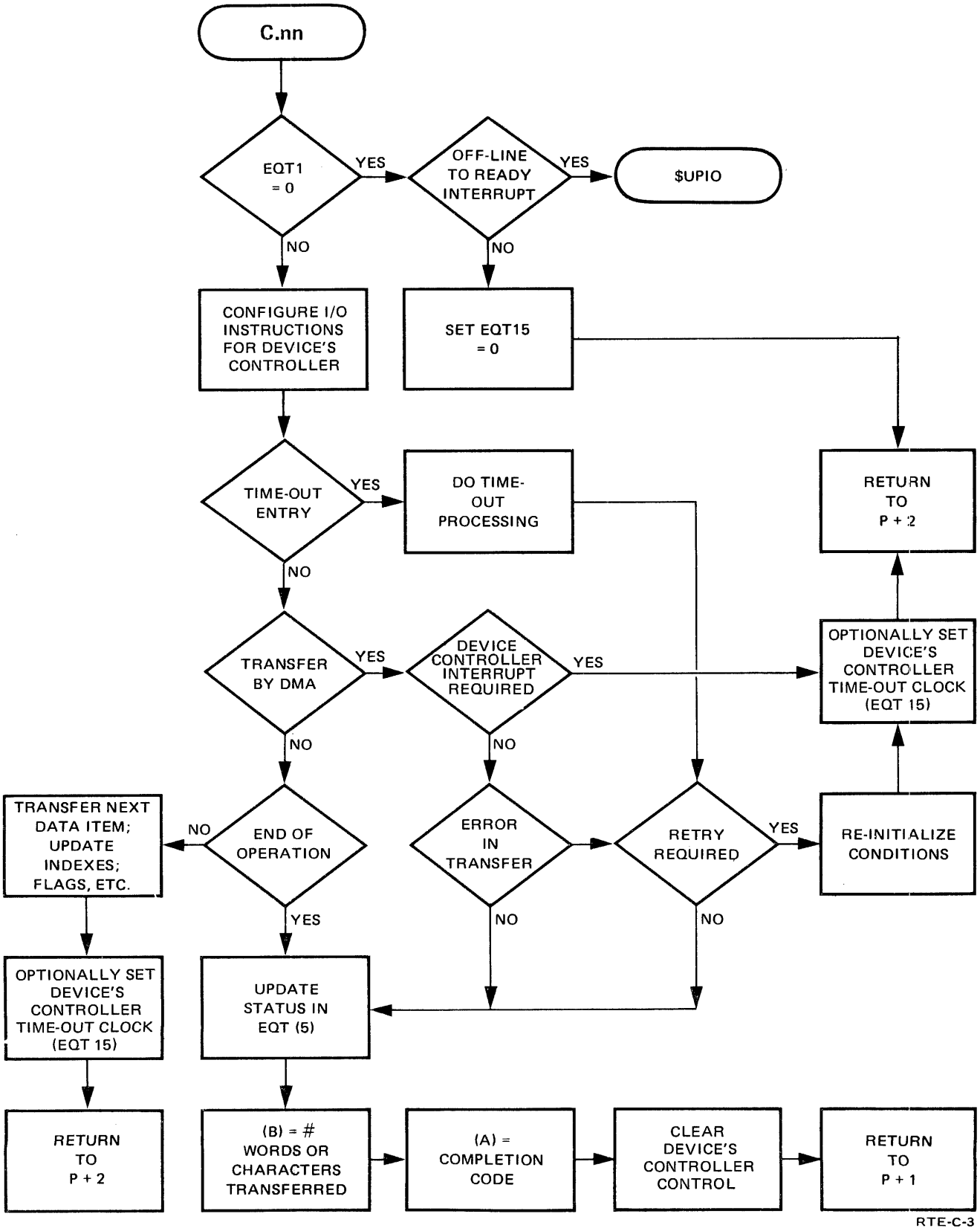


Figure 5-3. I/O Driver Completion Section

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 overrides 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 controller times out, the following actions take place:

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

I/O TO L #*x* E #*y* S #*z*

where *x* is the LU number, *y* is the EQT number, and *z* is the subchannel. The LU is set 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 LU is set to the down status, and bit 15 in the second word of the device's LU entry is set to one. An error message is printed; e.g.,

I/O TO L #*x* E #*y* S #*z*

c. The system issues a CLC to the controller'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 controller 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 controller 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 controller. If a time-out parameter is not entered, its value remains zero and time-out is disabled for the controller.

DEVICE CLEAR

If an I/O suspended program is aborted while waiting for a controller, the system clears the controller by sending a clear control request (00) to the driver. If the controller can be cleared without interrupt (i.e., immediately), the driver should return with A-Register = 2 or 4. If an interrupt is required, the driver should return with A-Register = 0. In this latter case the system will force a 1-second time-out for the controller. If the interrupt is not serviced before this time-out, the system will process it as in step "c" above. Note that the driver is not allowed to process this time-out.

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 controller is now ready, it may "up" itself as follows:

JMP $UPIO

The device's EQT and any of the EQT's downed LU's will be upped. If any requests are pending the system will call the driver at I.*xx*.

MAPPING SUBROUTINES FOR DRIVERS

There are two routines supplied with the RTE-III System that may be called by a driver to load the User Map to describe a certain program. $PVMP may be called by privileged drivers and $XDMP may be called by non-privileged drivers.

$PVMP SUBROUTINE (PRIVILEGED)

The $PVMP subroutine may be called by a privileged driver to load the User Map to describe a certain program. This routine is necessary only if the privileged driver needs to access a buffer in a user program. This routine is a type 8 module which means it is not re-entrant. A Type 8 module is loaded with each driver that calls it even though only one copy of the relocatable is input to RTGEN. Note that this routine should be used only by privileged drivers.

The calling sequence is as follows:

```
        EXT     $PVMP
        .
        .                       Normal privileged entry
                                Code (save registers,
                                etc)
        .
        .
        SSM     DMSST           Save DMS status at
                                interrupt
        LDA     MAPAD           Address of map
                                storage area
        IOR     SIGN            Set sign bit indicat-
                                ing store in memory
        USA                     Store user map in
                                memory
        LDA     IDADR           Get ID segment ad-
                                dress of program
        JSB     $PVMP           Go set user map
        SZA,RSS                 Check for error return
        JMP     ERROR
        UJP     CONT            Enable user map and
                                continue
CONT    .
        .                       Processing for privi-
                                leged interrupt
        .
EXIT    LDA     MAPAD           Address of map stor-
                                age area
        USA                     Restore original map
        JRS     DMSST RTN       Restore DMS status
        .                       at interrupt and
        .                       continue
RTN     .                       Restore registers and
        .                       memory protect
        .                       status then return to
                                point of interrupt
        MAPAD
MAPAD   DEF     MAP
MAP     BSS     32              Map save area
SIGN    OCT     100000
IDADR   BSS     1               Storage for ID seg-
                                ment address
DMSST   BSS     1               Storage for DMS
                                status at interrupt
```

$PVMP will check to see if the program is resident in memory. If it is not, the User Map will not be reloaded and the A-Register will be zero on return. If the program is resident, the User Map will be loaded and the A-Register will be non-zero on return. Note that any privileged driver using this routine must save the contents of the User Map before calling this routine and must restore the contents before returning to the point of interrupt.

$XDMP SUBROUTINE (NON-PRIVILEGED)

The $XDMP subroutine may be called by a non-privileged driver to load the User Map to describe a certain program. Since the system will enter the driver with the map enabled which describes the buffer of the current I/O call, this routine is necessary only for those drivers which must access another buffer which is contained in a user program. Note that this routine is intended for the use of non-privileged drivers only and is included as part of the system. The calling sequence is as follows:

```
        EXT     $XDMP
        .
        .                       Normal driver proc-
                                essing
        .
        RSA                     Get DMS status
        RAL,RAL                 Position current
                                status in upper bits
        STA     DMSST           Save status for later
        LDA     MAPAD           Address of map
                                storage area
        IOR     SIGN            Set sign bit indicat-
                                ing store in memory
        USA                     Store user map in
                                memory
        LDA     IDADR           Get ID segment ad-
                                dress of program
        JSB     $PVMP           Go set user map
        SZA,RSS                 Check for error
                                return
        JMP     ERROR
```

```
        UJP     CONT    Enable user map and
                        continue
CONT    .
        .               Process buffer under
        .               User Map
EXIT    LDA     MAPAD   Address of map
                        storage area
        USA             Restore original map
        JRS     DMSST   NXT Restore earlier
        .               DMS status and
NXT     .               continue
        .               Proceed with normal
        .               processing
        .
        .
MAPAD   DEF     MAP
MAP     BSS     32      Map storage area
SIGN    OCT     100000
IDADR   BSS     1       Storage for ID seg-
                        ment address
DMSST   BSS     1       Storage for DMS
                        status
```

The routine will check to see if the program is resident in memory. If it is not, the User Map will not be reloaded, and the A-Register will be zero on return. If the program is resident, the User Map will be reloaded and the A-Register will be non-zero on return.

Note that any driver using this routine must save the contents of the User Map before calling this routine and must restore the contents before exiting.

It is recommended that privileged drivers be designed for user communication through common or subsystem global. If this is done, the driver does not have the overhead of map switching; it simply saves and restores the state of the machine (including DMS status).

SAMPLE I/O DRIVER

The sample driver in Figure 5-4 demonstrates the principles involved in writing an I/O driver for the RTE-III 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.R. (OUTPUT) **


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

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 /M *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)

PRIVILEGED INTERRUPT PROCESSING

When a special I/O interface card is included in the system, RTE-III allows a class of privileged interrupts to be processed independently of regular RTE-III 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 the generation section for the exact specification procedure.

The special I/O card physically separates the privileged interrupts from the regular system-controlled interrupts. When an interrupt occurs the card has its flag set which enables the card to hold off non-privileged, lower priority interrupts. This means that the system does not operate with the interrupt system disabled, but in a hold-off state. Furthermore, the privileged interrupts are always enabled when RTE-III is running, and can interrupt any process taking place.

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 located (embedded) in the system area.

If a privileged driver 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 a special routine is used, a jump subroutine indirect (JSB xxx,I) instruction 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 the system are not aware of these interrupts but CPU time is lost because of them. For this reason the special routine must functionally be completely independent of the system (i.e., the routine can not use any of the system functions). CIC sets a software flag (MPTFL) in base page indicating the current status of the memory protect fence.

If MPTFL equals zero, memory protect was "ON" at the time of interrupt. Any special interrupt routine must restore the status of memory protect and the dynamic mapping system as described below before returning to the point of interruption by a JMP xxx,I instruction.

IF MPTFL equals one, RTE-III itself was executing when the privileged interrupt occurred, and memory protect was "OFF". The special routine must not restore memory protect in this case.

MEMORY ACCESS BY PRIVILEGED INTERRUPT

A privileged interrupt routine, whether embedded directly within the system or within a privileged driver, must save and restore all registers which are used (including index registers), restore memory protect to its original state (word MPTFL contains this status), and restore the status of the Dynamic Mapping System. If the privileged driver wants to access a buffer within a disc resident program, the User Map will have to be saved and restored. In addition, it will have to be loaded to describe the correct partition. However, if the driver limits its access to a buffer in either common or the subsystem global area, the driver may access that buffer while in the System Map if the generation option to include common and SSGA in the System Map was exercised. That is, if the program the driver is servicing puts a buffer in the common area, then the driver can access that data through the System Map which is enabled by the interrupt. Note that no standard HP software uses common; it is reserved for the user.

SPECIAL PROCESSING BY CIC

The Central Interrupt Control (CIC) module is entered on all normal (non-privileged) interrupts. CIC disables the entire interrupt system (including privileged), saves registers, issues a clear flag instruction (CLF) to the interrupt location, sets the memory protect "OFF" (MPTFL=1), and checks the DUMMY word. If the DUMMY word is zero, the hardware interrupt system is left disabled and normal processing continues. If non-zero, a set control instruction (STC) is issued to the I/O location specified (this assumes that the flag on the special I/O card is set). The STC holds off lower priority interrupts until the control flip-flop is cleared on the special card.

If DUMMY is non-zero, CIC also clears the control flip-flop on each DCPC channel to defer DMA completion interrupts, and enables the interrupt system (a NOP in the interrupt location for the special card causes its interrupts to be ignored). The DMA transfer itself is not affected, only the completion interrupt.

Interrupt processing continues and control is passed to a driver, timer routine, scheduler, or other appropriate executive module. Privileged interrupts can occur during this processing. RTE-III returns to user program processing through the interrupt return module, $IRT.

$IRT briefly disables the entire interrupt system. Each active DCPC channel is reset (STC) to allow it to interrupt. User register values are restored, and the memory protect system is inactivated. The User Map is enabled and control is transferred to the user program with the interrupt system on.

PRIVILEGED INTERRUPT ROUTINES

A privileged interrupt routine, whether embedded directly within the system or within a privileged driver, must save and restore all registers which are used (including index registers), restore memory protect to its original state, and restore the status of the Dynamic Mapping System. The privileged interrupt routine must not use any features or requests of RTE-III, or either DCPC 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 predefined locations in common. The starting address of the common region is available in base page. (See Appendix A.) A normal user program could, for example, be scheduled to run at periodic time intervals to scan and set indicators in common, and fill or empty buffers in that area.

SAMPLE PRIVILEGED DRIVER

The following discussion describes a sample privileged driver (see Figure 5-5), generalized to DVRXX, which is controlling a device operating in the privileged mode. The user buffer is in common.

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 controller'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's controller interrupts, the RTIOC overhead is circumvented because the privileged routine is entered directly.
- After each interrupt, if another data point is still required to satisfy the buffer length, the device's controller 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-III driver uses the EQT for all its temporary storage so that the same driver can be driving more than one device controller 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 controller. For each device controller 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 controller 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 controller 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 controller (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 controller 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 controller interrupts will be channeled to the P.XX subroutine instead of to RTIOC. The "JSB P.XX" instruction and its associated base page link are established via the small program "$JPXX" (see listing).

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

PRIVILEGED SECTION, P.XX. When the device controller interrupts, P.XX is entered as a result of the controller's trap cell modification. The system Map will be enabled by the interrupt.

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. It must also save the dynamic mapping system status at the time of interrupt.

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 controller for the next reading, disables the interrupt system, encodes the device controller, 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 controller, and sets up the driver for an immediate time-out. Before P.XX exits, it restores memory protect status and its flags, turns the interrupt system back on and restores the dynamic mapping system status.

COMPLETION SECTION, C.XX. The status of the device controller 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 reschedules the calling program and processes the controller request list as if it were a standard (non-privileged) controller.

ACCESS TO BUFFER IN USER AREA

If the sample privileged driver described above were written to access a buffer within the user program area instead of common, the following changes would be necessary: (The caution note on the sample also applies here, since information is stored in the driver).

1. I.XX – The ID segment address of the calling program would have to be saved. This would be done as follows:

```
I.XX    STA     SCODE       Save select code
        LDA     1717B       Get ID Segment
                            address
        STA     IDADR       Save it
```

2. P.XX – In addition to saving and restoring the DMS status, the User Map would have to be saved, reloaded to describe the user, and restored on exit. This would be done as follows:

```
P.XX
        .
        .
        .
P.XI    SSM     DMSTS       Save DMS status
        LDA     MAPAD       Get address of map
                            storage area
        IOR     SIGN        Set sign bit indicating
                            store to memory
(37μSEC)USA                 Save user map
        LDA     IDADR       Get ID segment ad-
                            dress
(100μSEC)JSB    $PVMP*      Call routine to set
                            map
        UJP     CONT        Enable user map and
                            jump to CONT
CONT    LDA     MPTFL
        .
        .
        .
EXIT1   LDA     MAPAD       Get address of map
                            storage area
        USA                 Restore user map
        LDA     EOSV
        .
        .
        .
MPADR   DEF     MAP         Address of save
                            Map area
IDADR   BSS     1           Save ID segment
                            address
MAP     BSS     32          Save map area
DMSST   BSS     1           Save DMS status
SIGN    OCT     100000      Sign Bit
```

The subroutine $PVMP is a Type 8 subroutine and is loaded with each driver that accesses it. It is not re-entrant.

The times given in the example are approximate.

```
 T=00004 IS ON CR00002 USING 00023 BLKS R#0203

ASMB,R,L,T,8
*
*
*  DRIVER WITH PRIVILEGED INTERRUPT
*
*
       NAM DVRXX
       ENT I.XX,P.XX,C.XX
       EXT $JPXX
*
*
*
*  CALLING SEQUENCE:
*      JSB EXEC       CALL EXEC
*      DEF *+5        RETURN POINT
*      DEF RCODE      REQUEST CODE
*      DEF CONWD      CONTROL WORD
*      DEF BUFFR      ADDRESS OF BUFFER
*      DEF LENTH      LENGTH OF BUFFER
*
*
*
*
*  CAUTION:  THIS DRIVER WILL NOT WORK WITH MORE THAN
*      ONE SUBSYSTEM.  IF MORE THAN ONE SUBSYSTEM
*      EXISTS IN A SYSTEM, BOTH DVRXX AND $JPXX MUST
*      BE RE-ASSEMBLED WITH ALL THE NAMES CONTAINING
*      'XX' CHANGED TO SOME OTHER NUMBER.  THEN ONE
*      DRIVER PER SUBSYSTEM MUST BE PUT INTO THE SYSTEM
*      AT GENERATION TIME.
*
*  INITIATION SECTION
*
I.XX   NOP
       STA SCODE      SAVE SELECT CODE
       LDA EQT6,I     REQUEST CODE TO A
       AND M77
       CPA =B1        READ REQUEST?
       JMP *+3         YES
REJCT  CLA,INA         NO - ERROR
       JMP I.XX,I     REJECT RETURN
FIRST  RSS            CONFIGURE FIRST
       JMP INIT        TIME ONLY
*
       LDA SCODE
       IOR LIA        CONFIGURE
       STA IO0         IO INSTRUCTIONS
       .
       .
       .
       LDA $JPXX      SET TRAP CELL TO
       STA SCODE,I     JSB P.XX
       LDA EQT4,I     CLEAR EQT4 BIT12
       IOR BIT12       TO ALLOW NORMAL
       XOR BIT12       TIME OUT.
       STA EQT4,I
       LDA EQT15      SAVE EQT15
       STA EQ15        AND EQT4 ADDRESSES
```

Figure 5-5. Sample Privileged I/O Driver

```
        LDA EQT4        FOR LATER.
        STA EQ4
        CLA             SET SO AS NOT TO
        STA FIRST        CONFIGURE AGAIN
*
INIT    LDA EQT8,I      NUMBER OF CONVERSIONS TO A
        CMA,INA         NEGATE FOR
        STA CVCTR        CONVERSION COUNTER
        SSA,RSS         REJECT IF
        JMP REJCT        NUMBER <=0
        LDA EQT7,I      SAVE DATA BUFFER
        STA DAPTR        ADDRESS FOR P.XX
        JSB READ        START A READING
IO1     STC IO,C        ENCODE DEVICE
        JMP I.XX,I      RETURN
*
READ    NOP             ROUTINE CONTAINING
        .                CONFIGURED IO
        .                INSTRUCTIONS TO
        .                SET UP THE DEVICE
        JMP READ,I       TO INITIATE ONE READING
*
*  PRIVILEGED INTERRUPT ROUTINE
*
P.XX    NOP
        CLF 0           TURN OFF INTERRUPTS
        CLC 6           TURN OFF
        CLC 7            DMA INTERRUPTS
        STA ASV         S
        STB BSV          A
        ERA,ALS           V
        SOC                E
        INA
        STA EOSV         REGISTERS
        STX XSV         SAVE INDEX
        STY YSV          REGISTERS
P.X1    SSM DMSTS       SAVE DMS STATUS
CONT    LDA MPTFL       SAVE MEMORY
        STA MPFSV        PROTECT FLAG
        CLA,INA         TURN OFF MEMORY
        STA MPTFL        PROTECT FLAG
        STF 0           TURN ON INTERRUPTS
*
        .               LOAD IN DATA
        .                FROM DEVICE
        .                VIA IO INSTRUCTIONS
*
        STA DAPTR,I     STORE IN DATA BUFFER
        ISZ CVCTR       LAST CONVERSION
        RSS              NO
        JMP DONE         YES
        ISZ DAPTR       SET UP FOR
        JSB READ         NEXT CONVERSION
        CLF 0           TURN OFF INTERRUPTS
IO4     STC IO,C        ENCODE DEVICE
*
EXIT    LDA MPFSV       WAS MEMORY
        SZA              PROTECT ON?
        JMP EXIT1        NO, FORGET DMA'S
        LDB INTBA       TURN
```

Figure 5-5. Sample Privileged I/O Driver (continued)

```
        LDA B,I         DMA'S
        SSA              BACK
        STC 6             ON
        INB                IF
        LDA B,I             THEY
        SSA                  WERE
        STC 7                 ON
EXIT1   LDA EOSV        RESTORE
        CLO              E AND
        SLA,ELA          O
        STF 1              FLAGS
        LDB BSV         RESTORE B
        LDX XSV         RESTORE INDEX
        LDY YSV          REGISTERS
        LDA MPFSV       RESTORE MEMORY PROTECT
        STA MPTFL        FLAG IN SYSTEM
        SZA             MEMORY PROTECT ON?
        JMP EXIT2        NO
        JRS DMSTS EX1
  EX1   LDA ASV           YES, RESTORE A
        STF 0           TURN ON INTERRUPTS
        STC 5           SET MEMORY PROTECT
        JMP P.XX,I           RESTORE DMS
*                             STATUS AND
*                             RETURN
EXIT2   LDA ASV         RESTORE A
        STF 0           TURN ON INTERRUPTS
        JRS DMSTS P.XX,I   RESTORE DMS
*                             STATUS AND
*                             RETURN
*
DONE    CLF 0           TURN OFF INTERRUPTS
IO7     CLC IO          TURN OFF DEVICE
        CCA             SET TIME OUT FOR
        STA EQ15,I       ONE TICK AND SET
        LDA EQ4,I        BIT12 IN EQT4 SO
        IOR BIT12        RTIOC WILL CALL
        STA EQ4,I        C.XX ON TIME OUT.
        JMP EXIT        GO TO EXIT
*
*
*  COMPLETION SECTION
*
*
C.XX    NOP
        CLA             SET UP FOR NORMAL RETURN
        LDB EQT8,I      TRANSMISSION LOG TO B
        JMP C.XX,I      RETURN
*
*
*  CONSTANTS AND TEMPORARIES
*
*
SCODE   OCT 0
CVCTR   OCT 0
        .
        .
        .
LIA     LIA 0
M77     OCT 77
DAPTR   DEF 0
```

Figure 5-5. Sample Privileged I/O Driver (continued)

```
ASV    OCT 0
BSV    OCT 0
EOSV   OCT 0
MPFSV  OCT 0
EQ4    NOP
EQ15   NOP
BIT12  OCT 10000
*
*
*  SYSTEM COMMUNICATION AREA
*
.      EQU 1650B
INTBA  EQU .+4
EQT4   EQU .+11
EQT6   EQU .+13
EQT7   EQU .+14
EQT8   EQU .+15
EQT15  EQU 1774B
XA     EQU .+49
XB     EQU .+50
XEO    EQU .+51
MPTFL  EQU .+80
A      EQU 0
B      EQU 1
       END
```

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

00001  ASMB,R,L,B
00002          NAM $JPXX
00003          ENT $JPXX
00004          EXT P.XX
00005  $JPXX   JSB P.XX
00006          END
```

Figure 5-5. Sample Privileged I/O Driver (continued)

SECTION VI
RTE SYSTEM INSTALLATION

The RTE-III system is initially configured with the system off-line generation program RTGEN. For instructions on using the RTE On-Line Generator see *RTE-II/III On-Line Generator Reference Manual* (92060-90016). Configuration is accomplished by the user entering answers in response to questions from RTGEN. This section describes the steps necessary to produce such answers.

To facilitate system generation, this section has been divided into three parts.

- Part 1 – Planning. This part contains directions for planning or laying out an RTE-III System. The overall structure of the system, including I/O and memory configuration, is planned with the aid of worksheets. It is imperative that the user be familiar with Section I of this manual before planning the system.

- Part 2 – Procedure. This part converts the plans from Part 1 into a procedure used to prepare generator responses to configure the system. The procedure is outlined on worksheets in the form of questions from RTGEN and responses by the user.

- Part 3 – Operation. This part presents the operation of the generator including features making the process more convenient. A sample generator is presented and discussed step-by-step (the sample is based on the generation listing in Appendix C). The last step in this part is the initialization (or start-up) of the new system.

DESCRIPTION

RTE-III is a multiprogramming system using partitions which are numbered contiguous blocks of memory the size and number of which are fixed during generation. The basic purpose of the generation is to build a system structured as shown in Figure 6 - 1. During the generation, various program modules are loaded and questions answered. The memory resident parts of the system are constructed and stored on the disc. The remainder of memory is divided into partitions for disc resident programs and these programs are relocated and saved on the disc to be swapped into memory when needed. The relocatable subroutine library is saved on disc for use by programs relocated during normal system operation.

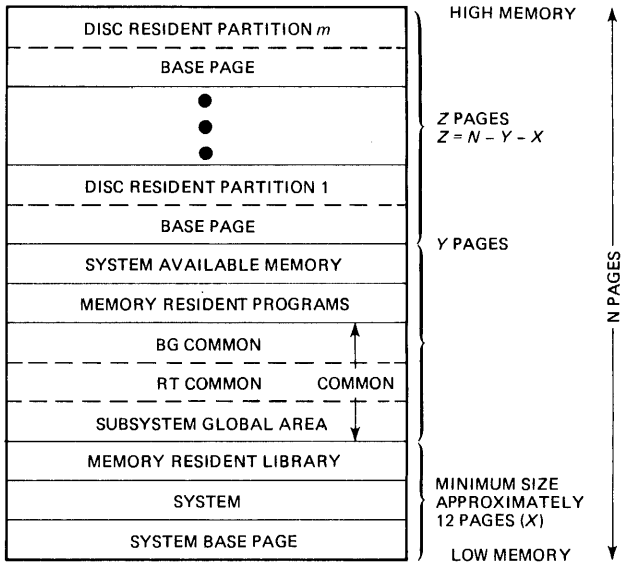


Figure 6-1. Physical Memory Allocations

Be aware that certain software subsystems may have specific requirements when included in the system generation. Options in such areas as spooling, measurements, communications, and multiple terminal operation may place specific requirements on I/O configuration, buffer space, etc. As an example, refer to the Batch and Spool Manual (HP Part No. 92060-90013) when planning for that subsystem.

PART 1
Instructions for Planning RTE

This planning part has been divided into three major areas as follows:

- Disc Planning – Disc tracks are grouped together to form subchannels. A scratch (work) area is set aside to temporarily hold programs during the generation process. Tables 6-1 and 6-2 provide the necessary worksheets.
- I/O Planning – Peripheral device I/O cards are assigned priorities, logical unit numbers are assigned, and tables are built that effect communication between the devices and the system. Table 6-4 provides the I/O worksheet.
- Memory Configuration – The physical and logical organization of memory is discussed and planned. Memory protection options are presented.

It is recommended that all the worksheets be duplicated. The copies can then be used for planning the system leaving clean copies in the manual for future use.

DISC PLANNING

RTE-III is a disc-based operating system where the disc provides the primary storage area for the following items:

- The configured operating system
- Relocated disc resident programs
- Relacatable library modules
- Temporary storage for programs (source for editing, relocatable output of assembler, etc.)
- User files

Disc storage is managed in terms of contiguous groups of tracks called subchannels (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-III 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, "Multiple Disc Controllers," and also in Appendix B.

SYSTEM/AUXILIARY SUBCHANNELS

The RTE-III System disc tracks are those for which RTE-III controls and maintains a track usage table. Programs may obtain and release tracks from this area using calls to the executive. System tracks include all tracks on the system subchannel (LU2) and the auxiliary subchannel (LU3). 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 (e.g. 203 for a 7900 disc). 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 main differences between a system and auxiliary subchannel are:

- The configured system (including the memory resident system, the relocated disc resident programs, and the relocatable library), is stored on the system disc.
- The auxiliary disc is optional.
- Most program swapping takes place on the auxiliary 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-III System that are assigned to LU2 or LU3 (system or auxiliary subchannel) must be unique to that RTE-III System. Remaining tracks on the disc can be assigned to other systems.

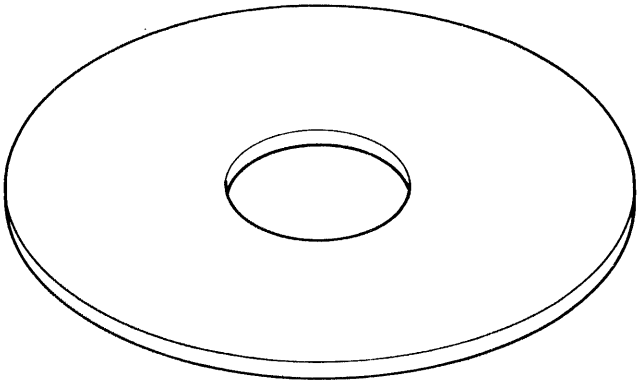
PERIPHERAL SUBCHANNELS

Disc subchannels other than system and auxiliary are classified as "peripheral" and must be assigned logical unit numbers above 6. Tracks on these peripheral subchannels are not subject to the RTE assignment and release mechanism; however, they may have the same protection. 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 1204 tracks.

Table 6-1. HP 7900 Moving Head Disc Worksheet

SUBCHANNEL 1

REMOVABLE



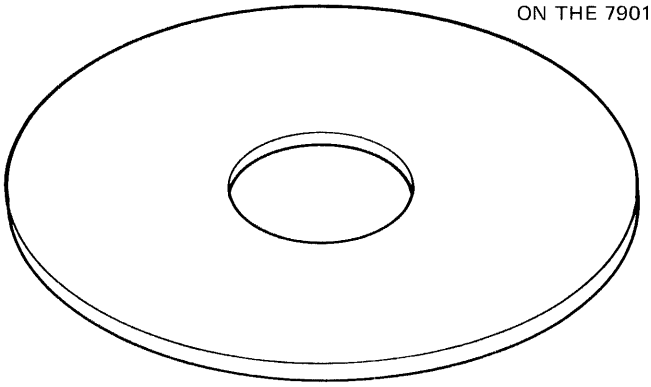
NO OF TRACKS AVAILABLE _____

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)

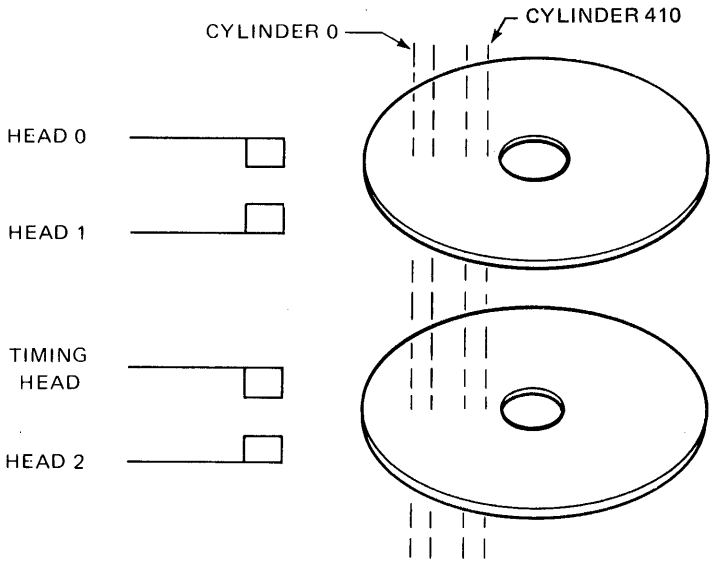
AUXILIARY SUBCHANNEL NUMBER _____ (LOGICAL UNIT 3)

SCRATCH SUBCHANNEL NUMBER _____

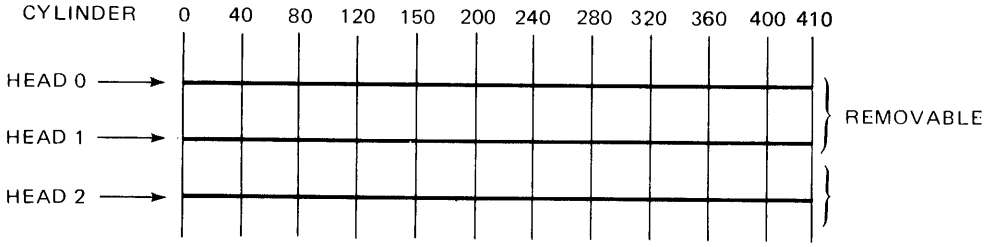
START SCRATCH (I.E. 1ST TRACK = 0) _____

Table 6-2. 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 ? (✓)						

HP 7900 DISC CONFIGURATION

The HP 7900 Disc Drive is a single unit that contains two discs; 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 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 platter 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. Refer to Table 6-1 and fill in the blanks according to the following instructions.

Determine the number of tracks available and starting track number for each subchannel, and fill in the blanks on the worksheet. Note that the maximum number of tracks available per subchannel for the 7900 is 203. The moving head Basic Binary Disc Loader (BBDL) will boot a system on a 7900 disc only if it starts at physical track 0 on subchannel 0 or 1. Locating the system tracks anywhere else will require that a paper tape bootstrap be punched during generation and used each time the system is initialized.

Determine which subchannel will be the system and which subchannel the auxiliary (if any). Fill in the appropriate blanks on the worksheet.

Refer to the heading, "Multiple 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 8 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 (3 heads) and 411 cylinders, giving 1,233 tracks. Refer to Table 6-2 for a pictoral diagram of the drive showing heads and cylinders.

The purpose of the following discussion is to configure each disc into subchannels. Each subchannel 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 fixed 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, bear in mind that the ultimate goal is a logical unit number referencing a group of disc tracks.

When filling in the worksheet on Table 6-2 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 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 since 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 original surface when necessary. For example, a subchannel beginning with head 1 and using 2 surfaces will use head 1, 2 and 1 repeatedly, and in that order. Note that any subchannel using 3 surfaces must start on head 0.

- Spare tracks. Some tracks on a disc surface may be unusable. When such a track is encountered, 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 their 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. A subchannel or complete disc might later be copied to another disc where bad tracks are encountered, and all data would not "fit" without sufficient spares.

- 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 boot a system on a 7905 disc only if it starts at 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 6-2, 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. Set the switch to the appropriate number and then write the 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 two surfaces. This makes a total of 264 tracks which is 132 cylinders. Note that the example on Table C-1 has all 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 6 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 the auxiliary (if any) and check the appropriate boxes.

MULTIPLE DISC CONTROLLERS

The RTE-III 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 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).

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.

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 area. (1) The area must be large enough to accomodate all the relocatable modules, otherwise, an ERR17 will occur, and (2) the area cannot be so large that the system area will prematurely overflow into it (ERR38) during the disc loading phase. The ERR38 can occur when the scratch area is located on the system subchannel and the absolute system is built upwards, toward, and into the relocatable modules, overlaying the latter before they have been converted into absolute code. Due to this possibility, it is recommended that the scratch area not be located on the system subchannel.

Determine which subchannel will provide the scratch area and indicate on the appropriate disc worksheet. If the scratch area is located on the system subchannel, it is recommended that the entry for "start scratch" be zero (0).

This entry causes RTGEN to start the scratch area at the midpoint of the available disc area. (Note that this default occurs only when scratch is located on the system subchannel.) If either of the two error codes (ERR17 or ERR38) are experienced during RTGEN, use the data in Table 6-3 as a guide in adjusting the scratch area. A rule-of-thumb formula for determining the approximate number of 64 word sectors a user written program will occupy is as follows:

$$\#\text{ of 64 word sectors} = \frac{x}{33}$$

where

x = number of words of memory the program occupies.

NOTE

Table 6-3 and the formula are only approximate guides to be used as an aid if using the RTGEN scratch default does not work or if difficulty is experienced in estimating some other starting point for scratch.

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.

Table 6-3. Approximate Number of 64-Word Sectors Required to Store RTE-III in Relocatable Format

NAME	64-WORD SECTORS
Executive Software	210
System Library	50
ASMB	210
XREF	60
FORTRAN	350
RTE/DOS FORTRAN IV Compiler	470
RTE/DOS FORTRAN IV 10K Compiler	360
ALGOL	180
Interactive Editor	70
Loader	150
RTE/DOS Relocatable Library	240
RTE/DOS FORTRAN IV Library	290
RTE/DOS HP FORTRAN Formatter	45
RTE/DOS Plotter Library	80
Drivers	Allow 15 sectors per driver.

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 number 10).

INPUT/OUTPUT PLANNING

Input/output locations in all HP 2100 series computers have the same sequence of priority addresses: the highest priority address is the lowest numbered select code (I/O location). The octal select codes start at octal 10 and continue upward toward 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 devices are assigned the lowest. The one exception to the direct memory access 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 by step numbers to the I/O Configuration Worksheet in Table 6-4. 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:

octal select code / subchannel	20	21
0		
1		
2		
3		

This method of noting subchannel numbers will facilitate assigning logical unit numbers later in the table. Refer to the HP 7905 Disc worksheet for applicable subchannel numbers.

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.

c. Assign the TBG I/O card.

d. Assign the moving head disc controller I/O cards.

e. Assign all devices that do not use direct memory access in order of decreasing speed.

NOTE

There will be occasions when a device uses direct memory access 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 direct memory access in order of decreasing speed.

g. If an I/O extender is required and the extender does not have direct memory access capability, the order of steps "e" and "f" can be reversed so that all direct memory access 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 disc subchannels for LU2; include LU3 if applicable. Any remaining disc subchannels can be assigned logical unit numbers above 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.

Table 6-4. I/O Configuration Worksheet

INPUT/OUTPUT CONFIGURATION WORKSHEET

GENERATION NUMBER ____________ DATE ____________ PREPARED BY ____________

	SC / SUB	10																													
STEP 1	I/O INTERFACE CARD NAME																														
	STD. LOGICAL UNIT NOS.																														
STEP 2	1 SYS. TTY																														
	2 SYS. MASS STORAGE																														
	3 AUX. MASS STORAGE																														
	4 PUNCH OUTPUT																														
	5 INPUT																														
	6 LIST OUTPUT																														
STEP 3 →	7_{10} to 63_{10}																														
STEP 4 →	DVR IDENT. (DVRxx)																														
STEP 5 →	DMA REQUIRED (D)																														
STEP 6 →	EQT ENTRY NO.																														
STEP 7 →	BUFFERED OUTPUT (B)																														
STEP 8 →	TIME OUT (T)																														
STEP 9 →	EXTENDED EQT																														
	OCTAL SELECT CODE / SUBCHANNEL	10																													

TPRTE-18

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 7900 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: DIRECT MEMORY ACCESS

Write in a large "D" for direct memory access 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 DCPC channel to themselves when required. In those cases, do not assign direct memory access. 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 DCPC 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 recommened 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 Spool Monitor Driver DVS43 will use the EQT extension. Values will be assigned later on the configuration worksheet.

MEMORY CONFIGURATION PLANNING

RTE-III, as described in Section I, provides the capability of addressing physical memory configurations of up to 1024K words. This portion of the planning part describes most of the considerations you must make when dividing up physical memory, setting up partitions, establishing memory protection, and actually loading programs. This material is provided for both reference and planning purposes to help the user. Some actual inputs to the generator will depend on the user analyzing the data printed out by the generator to that point, and making his decision based on that hard data with the aid of the considerations presented here.

PHYSICAL MEMORY

Physical memory is organized as shown in Figure 6-1. The organization is fixed although relative sizes of the areas will depend on installation needs. Some areas (e.g., common) will not exist in all systems. The user determines the size of system available memory, size of each partition, the size of common, and the size and composition of the resident library and memory resident program area.

MEMORY SIZE – The size of physical memory depends on the hardware supplied. RTGEN can configure a system from 32 to 1024 pages long.

SYSTEM BASE PAGE – The system base page contains the system communication area and is used by the system to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, etc. System and library links, memory resident program links, and trap cells are also located on the system base page. The base page links for memory resident programs are not accessible by disc resident programs and therefore may not be shared. System and library links and the system communication area are available to all programs for read-only access.

The system communication area is fixed. The size of the system links area varies with the number of page crossings which cause indirect links to be generated on base page (current page linking can reduce the number of base page links used; see Part 3).

After the assignment of I/O interrupt locations (see Input/ Output Planning), the user has no direct control over the allocation of the base page area. Linkages are allocated as needed during the generation. If the base page linkage area overflows an error message is given and the user must delete one or more programs from the memory resident area of the system. As an aid in generation, RTGEN will optionally trace the allocation of links, program by program.

SYSTEM AND LIBRARY AREAS – These two areas become a part of every program's logical address space (see Figure 6-2). Since each program is limited to 32K addressability, the size of this area directly reduces the area available for user programs, system available memory, and memory resident program area.

The system area contains type 0 system modules (e.g., RTIOC, SCHED, EXEC) and drivers plus tables. The size of the system area is indirectly controlled by the number of I/O devices configured (i.e., table sizes and drivers).

The memory resident library area contains those re-entrant or priviledged library routines which are used by the memory resident programs (type 6) or which are force loaded (type 14) at generation time. Placing a module in this area means it doesn't need to be appended to programs that call it, but it is subject to special design constraints so that two programs will not inadvertantly gain concurrent access. Refer to the section of the Relocatable Library when designing such subroutines.

COMMON AREA – This area is divided into three subareas: The Subystem Global Area (SSGA), the Real-Time Common Area, and the Background Common Area. Common's size is important to all memory resident programs (they can only use that part of 32K remaining after common), and to disc resident programs using one of the three subareas. Disc resident programs using one of the common subareas must "map" the whole common area, thus reducing the amount of logical address space left over for the program.

The Subsystem Global Area is used by HP subsystems and contains type 30 modules loaded sequentially. The modules are accessed by their entry point and not through common declarations.

The Real-Time Common Area defaults to the maximum size common declared by any main program "typed" to use it. Real-time programs use Real-Time Common as the default case. Background programs may use Real-Time Common if "reversed" common is specified.

The Background Common Area defaults to the maximum size common declared by any main program "typed" to use it. Real-time programs use Real-Time Common as the default case. Background programs may use Real-Time Common if "reversed" common is specified.

If a program (memory or disc resident) is to use common, the maximum size to be used must be declared in the main module. Subroutines and segments used by the program will access the same common as the main.

If desired, the size of the Real-Time and Background Commons may be increased during generation to accomodate future programs loaded on-line. Do not confuse these system-wide common areas with the local common area which may be specified for a program loaded on-line. The local common area is appended to the program (i.e., it will be in the program's partition), and is accessable only to that program, its subroutines, and its segments.

The common area may optionally be included in the System Map to aid privileged drivers. This makes common immediately accessable at interrupt. Refer to the part in Section V on privileged drivers.

MEMORY RESIDENT PROGRAM AREA – This area contains all type 1 programs and is loaded sequentially following common. It is recommended that the first word of this area be aligned on the first page boundary following common. The area skipped is then appended to Background Common. This alignment is desirable to protect the Memory Resident Program area. Refer to the heading, Memory Protection for more information. All memory resident programs must fall within the first 32K of physical memory. The last word of the last Resident Program must fall at or before 77677 (octal), leaving 64 words for operation of the loader. If this address is violated, ERR 18 is printed.

SYSTEM AVAILABLE MEMORY (SAM) – This is a temporary storage area used by the system for Class I/O, re-entrant I/O (refer to Section III), and automatic buffering The amount of SAM depends on specific applications. The lack of enough buffer space can cause temporary suspension or obortion of a program. Subsystem (communications, spooling, etc.) may place additional requirements on this area; refer to the appropriate manuals.

SAM may start immediately after the memory resident programs or be aligned at the next page. Alignment prevents accidental destruction of critical data by another program accessing the same page. Any words skipped due to alignment are wasted.

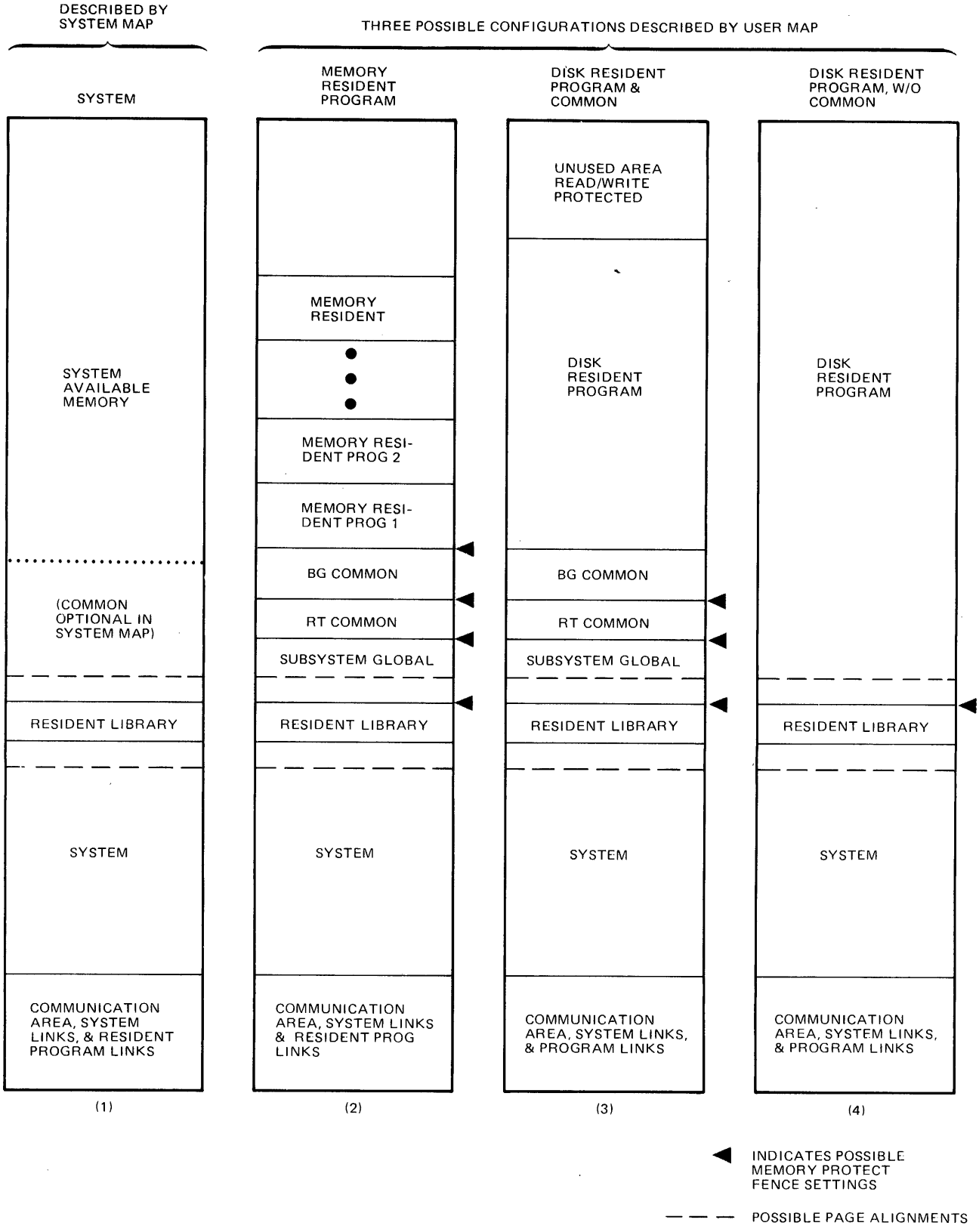


Figure 6-2. RTE-III 32K Logical Memory Configurations

SAM always ends at a page boundary where the first disc program partition starts. Therefore, its size defaults to the number of words between its starting address and the next page (between 1 and 1024 words). The recommended minimum is 1K words. The size limit is:

System + Library + SAM + (optionally)

Common $\leqslant$ 32 pages

System Available Memory size can be increased in 1K word increments by increasing the page number where the disc partitions start.

DISC PARTITIONS – The number of pages remaining after S A M must be divided up into distinct partitions (maximum of 64). Each partition should be at least two pages long, one page to be used as a base page and the remainder for the program. This includes subroutines, overlayable segments, and buffer/table space.

The size of a given partition depends on program needs. A Disc Resident program, out of its 32K of address space, usually has 13 to 16 pages taken up by the system and library area. Some programs use a common area which must be mapped. This may result in less address space for the programs depending on the size and location of the common area. Therefore, a useful partition will normally be between 2 and 20 pages long.

The generator reports the largest useful partition sizes for programs with and without common (including a base page for the program) to aid the user in determining partition sizes.

Partition size requirements for each program relocated are also reported; however, some programs may require additional pages for buffer area as discussed under the heading "Disc Program Size Considerations." It may not be possible to completely plan partition sizes until this information is reported by generator.

A program cannot be dispatched for execution unless a partition of sufficient size is defined and available (not reserved for the exclusive use of other programs).

The user must determine the mix of Real-Time and Background partitions of appropriate sizes to suit his particular application and subject to available main memory. Two classes of partitions prevents competition for main memory between background programs (typically involved in program development of other non-time-critical applications) and Real-time programs. Note that the class of a partition does not imply any special attributes, but merely that programs of the same type may use that partition subject to exceptions noted below.

In some situations, placing all partitions in a single class may be best. This allows free competition for main memory between all disc programs, subject to program priority and size requirements.

Undesired competition for partitions can be prevented by assigning programs to specific partitions. This could, for example, keep a very small program out of a large partition. Assignment can cross class boundaries; a Real-time program can run in a Background partition, and vice-versa. (Such a program would still have all the attributes of a real time program).

DISC PROGRAM SIZE CONSIDERATIONS

Section I of this manual discusses program size. The generator reports the partition size required for each disc program loaded. This size includes a base page and is based on the length of the main program, subroutines loaded with the main, and the largest overlayable segment (if any).

Program size can be overriden during the generation, thus increasing the minimum size partition required. When the program is run, it may be given a partition larger than this minimum. To the program however, the "apparent" size of the partition (determined from the System Communication Area during execution) is still the minimum.

Some programs require additional space to dynamically construct buffer areas or symbol tables. Standard RTE programs needing this additional space are shown with their requirements in Table 6-5. During generation the user must modify the page requirements of any of these programs to be used. Size requirements for user-supplied programs may be overriden if necessary.

MEMORY PROTECTION

Memory protection between disc resident program partitions and between disc and memory resident programs is provided by the Dynamic Mapping System. A program cannot access a page not included in its logical memory either directly or through a DCPC transfer. Since many programs will not use all of the possible 32K logical area, unused logical pages above the program are READ/WRITE protected and do not necessarily have counterparts in physical memory.

A different form of protection is required for the system, library, and (optionally), common. The memory protect fence provides this protection by preventing stores and jumps to locations below a specified address. All possible fence positions are shown in Figure 6-2.

Table 6-5. Programs Requiring Buffer Space in Partitions

PROGRAM NAME	MINIMUM RECOMMENDED OVERRIDE (pages)	SUGGESTED OVERRIDE (pages)
EDITR	6	7 (Note 2)
ASMB	7 (Note 1)	10 (Note 3)
XREF	6 (Note 1)	10 (Note 3)
LOADR	8 (Note 1)	10 (Note 3)
ALGOL	9 (Note 1)	13 (Note 3)
FTN	6 (Note 1)	8 (Note 3)
FTN4 (small)	7 (Note 1)	9 (Note 3)
FTN4 (large)	11 (Note 1)	13 (Note 3)
FMGR	7	7 (Note 4)
RT2GN	12	12 (Note 5)
RT3GN	12	12 (Note 5)

Note 1: Running this program with this size partition will limit the size of the programs it can process. In some cases, however, experience may show that even small partitions will suffice.

Note 2: Limited to "Largest Addressable Partition" size printed during generation. Extra space increases size of two disc buffers thereby improving performance.

Note 3: Limited to "Largest Addressable Partition" size printed during generation. Extra space increases symbol table space thereby allowing larger programs to be processed.

Note 4: Extra space is used during a disc packing operation.

Note 5: Extra space for the generator virtual symbol tables increases the generator's speed.

The memory protect fence applies to the logical address space and addresses are compared to the fence before translation. If a disc resident program does not use any of the common areas, the memory protect fence is set at the bottom of the program area. Similarly, for a memory resident program not using common, the memory protect fence is set at the base of the entire memory resident area.

For programs using common, all of logical memory including common is mapped and the fence is set at one of three possible locations, depending on the portion of common being used. A hierarchy of protection is thereby established within common due to their physical locations. Background common is the least protected (any program using any common can modify it) and SSGA is the most protected (only programs authorized for SSGA access can modify it). Figure 6-3 expands the common area and shows these three fence settings as (a), (b), and (c).

Figure 6-3 also shows a potential problem area marked "?" which includes those words from the top of common to the next page boundary. This area could include one or more memory resident programs and/or part of System Available Memory. Any program using common could potentially destroy the contents of this area. Aligning the top of common at the next page boundary is a generation option that expands the size of background common while eliminating this problem. A similar option is available for the boundary between memory resident programs and system available memory.

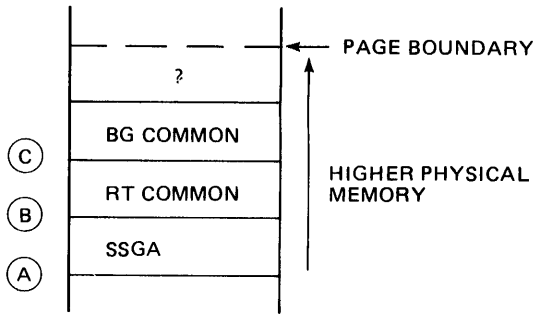


Figure 6-3. Memory Protect Fence Locations for Programs using Common.

PROGRAM LOADING

Program loading refers to RTGEN taking the relocatable modules from the scratch area, relocating them to absolute addresses in physical or logical memory, and storing them permanently on the system subchannel.

SYSTEM MODULES – These are type 0 modules (EXEC, RTIOC, SCHED etc.) and are loaded sequentially above the system base page. Base page links for these modules are allocated downward in the system base page below the system common area.

LIBRARY MODULES – These are type 6 and 14 (re-entrant, privileged, and force-loaded) and are loaded sequentially above the system and tables. Base page links for these modules are allocated downward in the system base page below the system links.

MEMORY RESIDENT PROGRAMS – These programs are sequentally loaded above the common areas. Base page links for these programs are allocated upward in the system base page starting at FWA BP LINKAGE (established by the user) above the I/O interrupt locations.

DISC RESIDENT PROGRAMS WITHOUT COMMON – These programs are relocated into logical memory and stored on the disc. Each program starts at word 2 of the next available logical page after the end of the system and memory resident library. The first two words of the page are reserved to save index registers in the event the program is interrupted. Base page links are allocated upward from logical location 2. The highest available link address is the word before the lowest system/library link. These links are written on the disc and are referred to as the user base page. This user base page is swapped with the program into memory and placed into the first page of the selected partition.

DISC RESIDENT PROGRAMS WITH COMMON – These programs are treated the same as the disc resident programs without common. The only difference is that the program starts at word 2 of the logical page following the common area.

PRIVILEGED DRIVERS

Privileged drivers must be considered when doing the generation. Practically, the privileged drivers will have already been written according to the directions given in Section V. If the driver was written to use the common area then the generator question about privileged drivers accessing common will have to be answered YES, causing common to be included in the System Map. Otherwise, it is assumed that driver is performing its own mapping functions.

PART 2
Preparing Generator Responses

The plans in Part 1 and the procedures described in Part 2 aid the user in preparing responses to RTGEN questions. These responses are written on Table 6-6, a worksheet located at the end of this part. The user can then use the worksheet to enter the correct responses to the computer as the generation proceeds. As the user becomes more familiar with the system and generation procedure, a punched tape containing all the parameter inputs can be made up from the worksheets and then placed in the tape reader (either the teletype tape reader or high-speed photoreader). The generator will read such a tape automatically and operate at a much higher speed than if the responses were entered interactively through the system console.

The worksheet has been keyed to the text by step numbers for easy cross-reference between the two.

This part is organized in parallel with the "phases" executed by RTGEN during operation. Some phases do not require user responses, but have been listed for completeness. The phases include:

- Initialization – Disc areas are described and initialized. Various system parameters are entered.
- Program Input – All relocatable tapes are loaded into the system by the user.
- Parameter Input – The default characteristics of programs just loaded can be overriden. Entry point values can be modified. More system parameters are entered.
- System loading – system executive routines, drivers and user written system routines are relocated by the system to absolute memory locations.
- Table Generation – tables describing the I/O configuration are constructed.
- Program loading – Memory resident library routines and memory resident programs are relocated by the system. Common areas are constructed and disc resident programs are relocated by the system.
- Partition definition – Partitions for disc resident programs are defined. Program size requirements can be modified, and programs may be assigned to run in specific partitions.

INITIALIZATION PHASE

The first portion of Table 6-6 is divided into two parts– one for HP 7900/7901 initialization and one for HP 7905 initialization. Refer to the appropriate heading and fill in the blanks for the type disc drive you have.

During the initialization phase, RTGEN first requests information necessary to generate track map that defines disc subchannels. Once the track map is established, RTGEN goes on to request more information necessary to generate the system.

HP 7900/7901 DISC INITIALIZATION

The answers to the following steps can be obtained from Table 6-1, the HP 7900/7901 System 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

Many of the answers to the following steps can be obtained from Table 6-2, the HP 7905 System 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 6-2.

STEP 3 – The number of 128-word sectors per track is 48. This is the number of logical sectors per track and is the number of sectors on two sides of the platter on a 7900 disc, one side on a 7905.

STEP 4 – Fill in the blanks for the system and scratch subchannels from the table. If only one subchannel was defined in step 2, place a dash (i.e., does not apply) in system subchannel, auxiliary disc, auxiliary disc subchannel, and scratch subchannel blanks.

STEP 5 – The auxiliary disc question is filled in as follows:

- Enter NO if none.
- Enter YES if it is to be one of the subchannels specified in step 2.
- Enter number of tracks on the subchannel if it is to be on a different controller.

If YES is entered, fill in the subchannel number in the next blank. Note that the auxiliary subchannel may have no more than 256 tracks assigned to it.

STEP 6 – Fill in the select code of the Time Base Generator (TBG) card.

STEP 7 – Fill in the select code of the privileged interrupt card. The next question is answered as follows: YES if the common area is to be included in the System Map for access by privileged drivers. NO if not.

STEP 8 – The core lock questions are answered with a YES or NO. If YES is answered to either of these questions, it allows a program of the corresponding type to lock itself into memory and not be swapped. If the answer is NO then the program cannot be locked into memory. Refer to the PROGRAM SWAPPING CONTROL call in Section III.

STEP 9 – 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. For the HP 7900 Disc Drive, the transfer time is 25 milliseconds for each 3K words. For the HP 7905 Disc Drive, the transfer time is 16.7 milliseconds for each 6K words. To obtain an accurate figure tailored to memory size, program size, and disc type, refer to Figure 6-4, Swap Delay Graph. (Note that the graph takes track switching time into account.) Remember, the number selected here is applied to all swappable programs.

STEP 10 – MEM SIZE refers to the decimal number of pages in the system. Fill in the blank.

STEP 11 – 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 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 (LIBR) input device. 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 3 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

The parameter input device is either PT or TY.

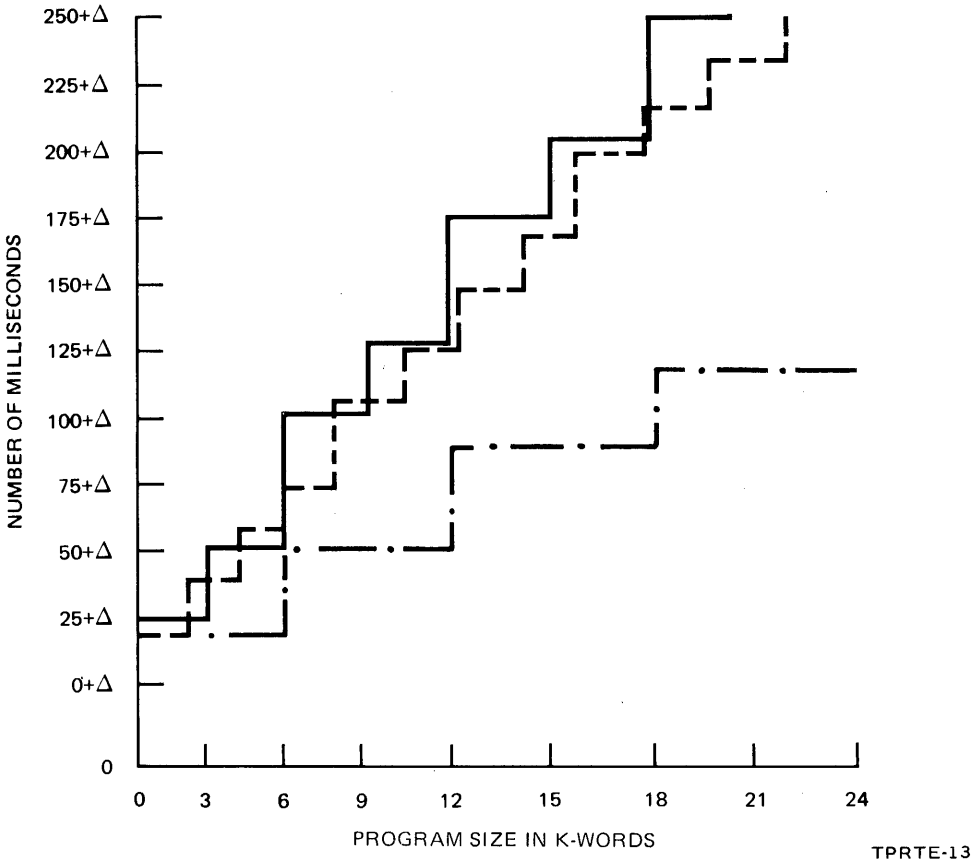
STEP 12 – The user has the option of initializing or not initializing the tracks on each subchannel. RTGEN automatically initializes tracks on the system, auxiliary, and scratch subchannels. If other subchannels are available RTGEN asks if they are to be initialized by subchannel number in ascending order. Write in YES or NO for each subchannel. If the answer is YES, RTGEN initializes the tracks on that subchannel and reports any defective tracks. (On a 7900 disc subchannel, only those tracks assigned to RTE in step 2 are initialized.) If the answer is NO, RTGEN does not initialize that subchannel.

NOTE

Any subchannels containing tracks 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).



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 6-4. Swap Delay Graph

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 disc controller to tracks set aside for that purpose in the initialization phase. If there are not enough spare tracks available, RTGEN issues the error message ERR43 and then restarts 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.

Defective tracks are reported as shown below:

	LOGICAL	CYL	HD	UNIT
BAD TRACK	xxxx	xxxx	x	x
SPARED TO	xxxx	xxxx	x	x

STEP 13 – The punch boot question is optional. If the computer is equipped with a built-in disc loader, the bootstrap tape may not be needed to initialize the system. This depends on where the first track of the system is located. Refer to Part 1. 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.

PROGRAM INPUT PHASE

At this point in the generation, all the relocatable modules are loaded into the system.

Due to the large number of tapes to be loaded, it is recommended that they be placed on a table in the following order. That way they are handy when the time comes to start loading.

Memory Resident System
I/O Drivers
Power Fail (DVP43)
System Programs written by the user
Multi-Terminal Monitor
Memory Resident Programs
Real-Time Disc 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
Editor
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 mains and their segments cannot be changed without losing their relationship to each other)

RTGEN has an additional feature that applies to memory and disc resident programs. During the Parameter Input Phase one program can be scheduled to execute automatically whenever the RTE-III 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 core from the disc and after the file manager has been scheduled for initialization. If more than one program is assigned for automatic scheduling, only the last one entered will be recognized.

STEP 14 – Write in the parameter records on the worksheet using the following general form:

name,type[,priority] [,execution interval]

Where:

name is the name of the program.

type is the program type code (the following are primary types):

0 – system program or driver
1 – memory resident
2 – real-time disc-resident
3 – background disc-resident
4 – not used
5 – background segment
6 – library, re-entrant or privileged (Note that these routines are relocated into the memory resident library if called by a memory resident program. If not called by a memory resident program, they become type 7.)
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 loaded in the relocatable library area of the disc.
14 – same as type 6 but automatically included in the resident library

The primary type may be expanded in some cases by adding 8 or 16 to the number. These expanded types allow such features as access to real-time common by background programs and access to SSGA. See Appendix I for a list of expanded program types.

priority is the program priority from 1 to 32767 with 1 the highest priority

execution interval is a list of six parameters (shown below) 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:

[res[,mult[,hour,min,sec,10msec]]]

res resolution code (0 to 4):

0 – no execution interval
1 – tens of milliseconds
2 – seconds
3 – minutes
4 – hours

mult execution multiple (0 to 4095); the resolution code gives the units for the execution multiple.

hour,min, sec, 10msec 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.

STEP 15 – The next set of blanks are for creating and modifying type 3 (absolute) and 4 (replace) entry (ENT) records. 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 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 loader will replace each reference to it with the octal (or decimal) value. For example:

.FMP,RP,105040

This would cause each JSB .FMP instruction (floating point multiply) to be changed to the microcode floating point multiply instruction (105040). Other floating point (or fixed point EAU) type instructions that could be entered are:

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			

Other uses include I/O configuration at load time, and configuring tables that are assembled as DEF statements to externals.

If you have loaded DVR32 into your system for the HP 7905 MH Disc, you may take advantage of the move words microcode by making the following entry point change:

.MVW, RP, 105777

If you want to protect the FMP peripheral cartridges from alteration by user programs, you may enter a change to an entry point when the generator issues the prompt CHANGE ENTS?. To protect these cartridges, specify:

$PDSK,AB,1

STEP 16 – The next item on the worksheet concerns blank ID segments. One blank ID segment is required for each program that will be loaded into the system on-line by the RTE-III relocating loader. If five ID segments are allocated, then only five additional programs can be loaded at any one time 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 disc resident program ID segment requires 29 words in the system memory 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 total number of program ID segments, including memory resident and disc resident programs, is limited to 256.

STEP 17 – The next item on the worksheet concerns blank background segment 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.

STEP 18 – The maximum number of partitions (64) is determined by dividing up the pages remaining after SAM. Directions for dividing up the memory into partitions is found in Part 1, Planning. Fill in the blank.

STEP 19 – The next item on the worksheet is for the address of the first word available on base page for memory resident program links (FWA BP LINKAGE). This address must be above the last used I/O select code. It may be convenient to defer filling in this answer until step 26 has been completed.

SYSTEM LOADING PHASE

The System Loading Phase requires no user inputs. During this phase the system starts relocating the programs previously entered.

TABLE GENERATION PHASE

The Table Generation Phase is the part of the generation where the user builds the EQT table, LU table, interrupt table, and satisfies other system parameters.

STEP 20 – 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).

STEP 21 – The next blank concerns a table (configured by the generator) called LU Mappings that cross reference real 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.

STEP 22 – 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).

STEP 23 – The next blank concerns current buffer limits. Setting 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 totals the lengths of all buffers for 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 and may reattempt their request. A suggested entry of 100 and 400 can be entered and later changed on-line with the BL command if desired. Note that programs with priorities of 1 through 40 are not suspended for buffer limit.

EQUIPMENT TABLE ENTRY (EQT TABLE)

STEP 24 – The first table to be completed is the Equipment Table. Most entries are located on the I/O worksheet and are transferred to this table. EQT number one should be the system disc and is either DVR31 for the 7900 or DVR32 for the 7905. 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., DVynn). The entry point names are four characters in length and start with either "I" (e.g., Ixnn for initiation section), or "C" (Cxnn 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 (extended EQT). The blanks are filled in as shown in the example in Figure 6-5.

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), starting at I/O initiation for the device before 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.

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 number of words of buffer space is appended to the EQT for the driver's 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, assume 6 files can be active at one time. The entries (referencing unused I/O slots) might be:

```
30,DVS43,X  =  18
31,DVS32,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 6-3) 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. An EQT entry specifying a non-existent (not loaded) driver results in generator error ERR 25.

DEVICE REFERENCE TABLE (DRT TABLE)

STEP 25 – 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 6-3) to obtain the EQT entry number, LU number, and subchannel number. Fill in the blanks as shown in Figure 6-6. LU0 (bit bucket) is a system mechanism that allows immediate I/O completion (i.e. data buffer is written to or read from a non-existent device) The first seven LU numbers are reserved for system devices as follows:

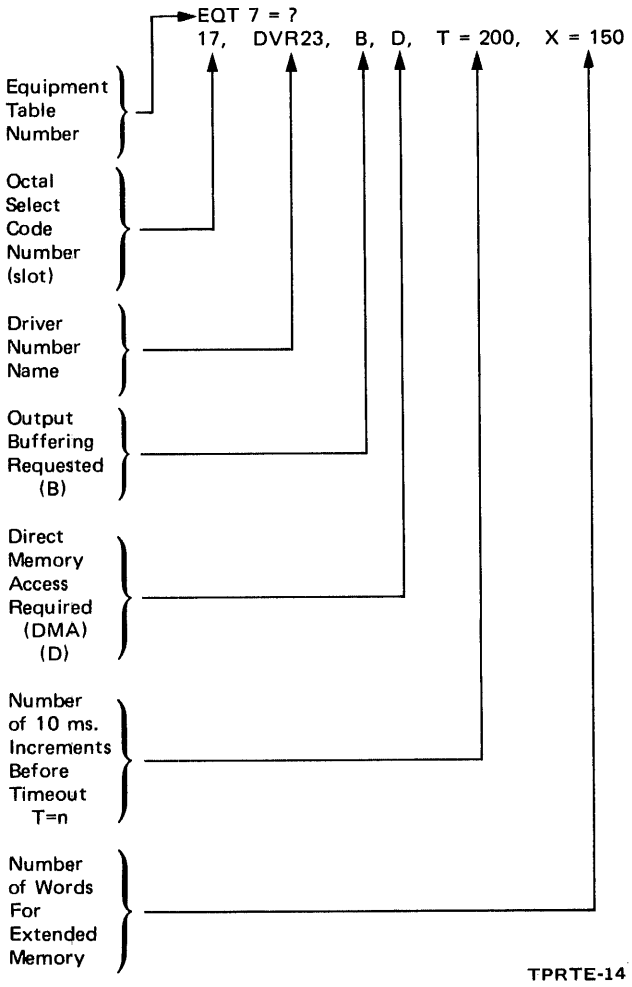


Figure 6-5. EQT Table Example

LU0 – Bit bucket (no entry required)
LU1 – System console
LU2 – System mass storage
LU3 – Auxiliary mass storage
LU4 – Standard output device
LU5 – Standard input unit
LU6 – Standard list unit
•
•
LU8 – Recommended for magnetic tape

Extra LU numbers can be assigned using EQT number zero and may be changed on-line to reference other EQT's as desired.

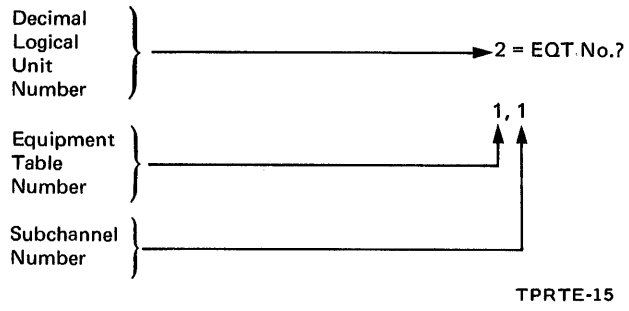


Figure 6-6. DRT Table Example

INTERRUPT TABLE (INT TABLE)

STEP 26 – 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 occurring on I/O slot 21 will be directed to EQT number one which has the address of DVR31 and that driver will be entered. The format for the entry is shown in Figure 6-7.

Refer to Figure 6-7 and construct the entries for the interrupt table as follows:

octal select code number is taken from I/O Worksheet (Table 6-3) in ascending order.

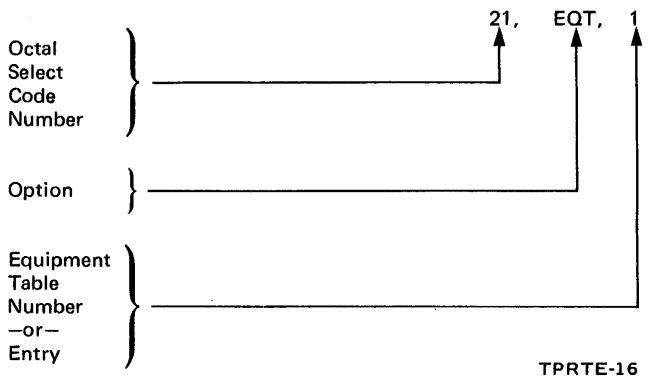


Figure 6-7. INT Table Example

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 the absolute octal value xxxxxx (instruction code) in the interrupt location. Do not place anything other than a JMP or JSB in the trap cell (see Appendix I).

The 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

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.

PROGRAM LOADING PHASE

Planning generation responses may be difficult beyond this point since some of the responses are based on data not yet known. Examining the worksheet shows that the generator relocates the library and SSGA area, printing out boundaries and requesting changes. The merits of changing these boundaries is discussed in Part 1, Planning. The remainder of this part may therefore be used as a reference in planning, and during generation.

STEP 27 – The library and SSGA areas are relocated. Real-time common default size is printed in decimal words. Enter the real-time common size desired in decimal words, or 0 (zero) for no change (only increases are allowed). The first word available (FWA) of real-time common is then printed.

STEP 28 – In this step the background default size is printed in decimal words. Enter the background common size desired in decimal words, or 0 (zero) for no change (only increases are allowed). The first word available (FWA) of background common is printed.

STEP 29 – The last word available (LWA) of background common is printed and the user is asked if he wants to align the end of common at the next page boundary (for protection of memory resident programs). Enter YES to align the end of common to the next page boundary. The updated last word address of background common is printed.

PARTITION DEFINITION PHASE

After answering the last question the generator relocates memory resident programs, real-time disc resident programs, and background disc resident programs. When these programs are all relocated, the generator print the name of each disc resident program and how many pages in memory they occupy (including base page). The largest addressable partitions, both with common and without common, are also listed. Partitions larger than these numbers can be declared, but the extra pages will not be accessable.

STEP 30 – The last word address of the memory resident program area is printed and the user is asked if he wants to align the end of the area to the next page boundary (for protection of the system available memory). Note that those words skipped are wasted. Enter YES to align the end of the memory resident program area to the next page boundary. If YES is entered, the generator automatically allocates one page of memory to the system available memory (SAM) area. If NO is entered, the generator allocates that area between the last word of the memory resident program area and the next page boundary, to SAM (this could be between 1 and 1024 words. The updated first word address of SAM is printed. The generator then prints the size of SAM.

STEP 31 – The first page available for disc partitions is printed and is the page boundary that ends SAM. The user may increase this number which increases the area (in pages) for SAM. Enter a new decimal page number or 0 (zero) for no change. The updated size of SAM, in decimal words, is then printed.

STEP 32 – The decimal number of pages of physical memory remaining are printed. This area must be subdivided into real-time and/or background partitions. The sum of partition sizes must equal the number of remaining pages. Enter each partition description using the format described below. The last entry is /E.

Format: *part #, size, class [,R]*

Where:

part#	is a number between 1 and maximum number of partitions (entered earlier in step 18). This number is the "name" of the partition.
size	is partition size in decimal number of pages (a partition must include enough pages for a program plus one page for the program's base page)
class	is RT for real-time or BG for background
R	is reserved flag; if specified, partition may only be used by programs specifically assigned to it (see step 34 below).

The order in which partition definitions are entered is up to the user. Partition numbers may be skipped if desired; however, pages will be assigned in order by partition number (lower numbered partitions get lower numbered pages). An example of defining the partitions is shown below:

1,15,BG (partition no. 1, 15 pages, background)

2,2,RT,R (partition no. 2, 2 pages, real-time, reserved)

STEP 33 – At this point the operator can modify disc resident program page requirements. The default size of each program was just prior to step 30 after the generator relocated the programs. This step allows the user to override the page requirements for those programs needing dynamic space for symbol tables or buffers. Refer to Table 6-5 for the standard RTE-III programs requiring a size override. Enter each disc resident program override using the format described below.

Format: *program name, pages*

Where: *program name* is the name of the program

pages is the decimal number of pages required to run this program (include one page for base page).

An example of entering the program override is shown below:

```
EDITR,8
```

The edit program EDITR is assigned 8 pages and will not be run in a partition with less than that number.

STEP 34 – The last step in the generation is assigning programs to run in a specific partition. Enter only those programs you wish to assign to a partition using the format described below with /E as the last entry.

Format: *program name, part #*

Where: *program name* is the name of the program

part# is a number between 1 and maximum number of partitions (entered earlier)

An example of assigning programs to partitions is shown below:

```
HENRY,1
```

The program HENRY will run only in partition number 1.

This concludes the operator inputs to the generator. The last thing the generator prints is the message that the system is stored on the disc and how many tracks and sectors it used.

Table 6-6. Generator Input Worksheet

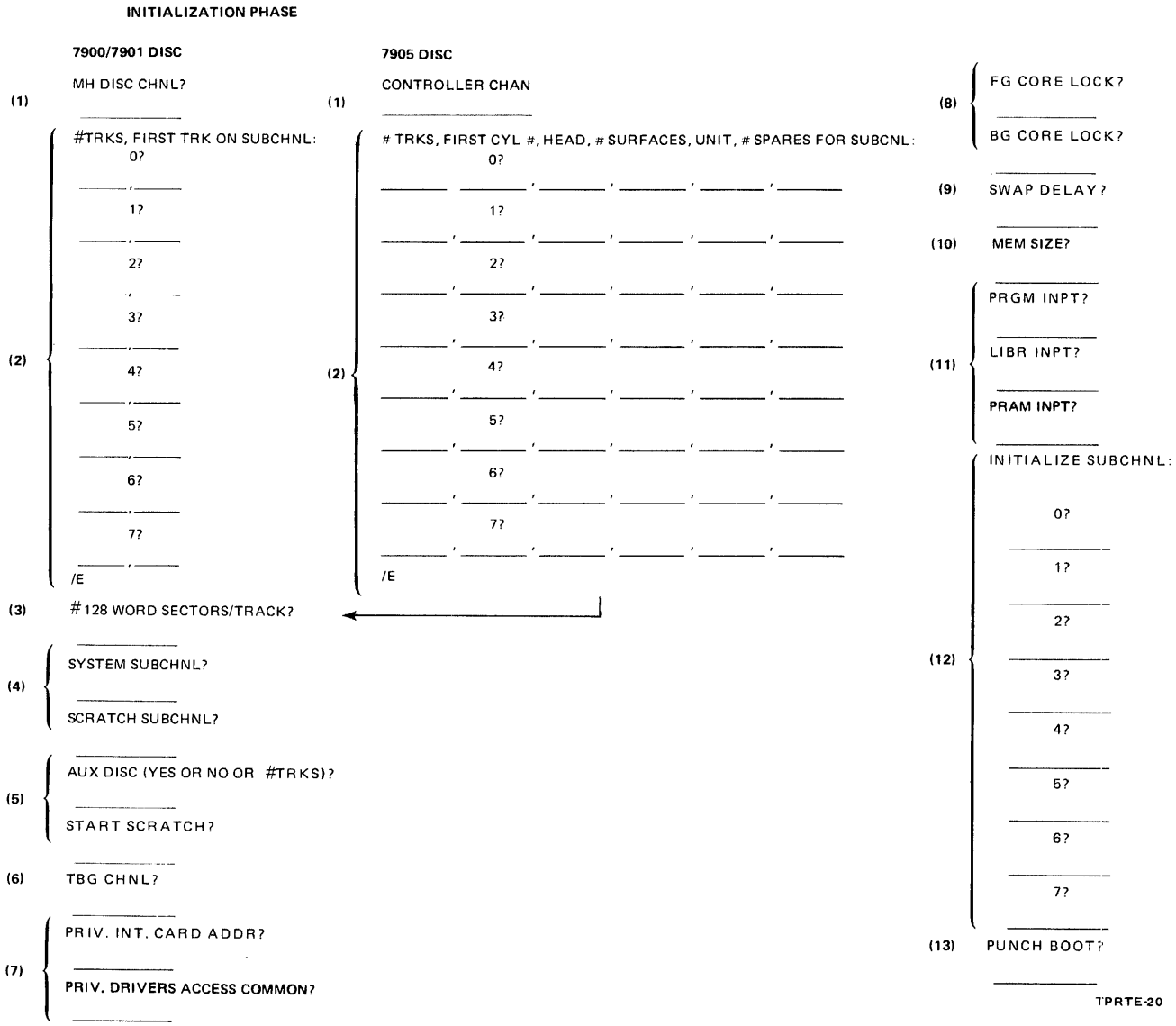


Table 6-6. Generator Input Worksheet (Continued)

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

(14) NAME, TYPE [,PR [, RES [, MULT [, HR, MIN, SEC, 10'S/MS]]]]

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

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

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

/E

(15) CHANGE ENTS?

_____,_____,_____

_____,_____,_____

_____,_____,_____

/E

(16) #OF BLANK ID SEGMENTS:

_____

#OF BLANK BG SEG. ID SEGMENTS?

(17) _____

MAX NUMBER OF PARTITIONS?

(18) _____

FWA BP LINKAGE?

(19) _____

SYSTEM LOADING PHASE
(NO USER INPUT)

TABLE GENERATION PHASE

*#OF I/O CLASSES?

(20) _____

* # OF LU MAPPINGS?

(21) _____

* # OF RESOURCE NUMBERS?

(22) _____

* BUFFER LIMITS (LOW, HIGH)?

(23) _____,_____

* EQUIPMENT TABLE ENTRY

(24)

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 6-6. Generator Input Worksheet (Continued)

(25)

```
* 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 #1
_____,_____
14 = EQT #?
_____,_____
15 = EQT #?
_____,_____
/E
```

(26)

```
* INTERRUPT TABLE
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
_____,_____,_____
/E
```

TPRTE-22b

Table 6-6. Generator Input Worksheet (Continued)

(27)
RT COMMON XXXXX
CHANGE RT COMMON?
_____

(28)
RT COMMON XXXXX
BG COMMON XXXXX
CHANGE BG COMMON?
_____

(29)
BG COMMON XXXXX
LWA BG COMMON XXXXX
ALIGN AT NEXT PAGE?
_____

LWA BG COMMON XXXXX

PARTITION DEFINITION PHASE

(30)
LWA MEM RESIDENT PROG AREA XXXXX
ALIGN AT NEXT PAGE?
_____

(31)
LWA MEM RESIDENT PROG AREA XXXXX
SYS AV MEM; XXXXX WORDS
1ST DSK PG XXXXX
CHANGE 1ST DSK PG?
_____

SYS AV MEM; XXXXX WORDS
PAGE REMAINING: XXXXX

(32)
DEFINE PARTITIONS
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
_____ , _____ , _____ , _____
/E

(33)
MODIFY PROGRAM PAGE REQUIREMENTS?
_____ , _____
_____ , _____
_____ , _____
_____ , _____
_____ , _____
_____ , _____
_____ , _____
_____ , _____
/E

(34)
ASSIGN PROGRAM PARTITIONS?
_____ , _____
_____ , _____
_____ , _____
_____ , _____
_____ , _____
_____ , _____
/E

SYSTEM STORED ON DISC
SYS SIZE: _____ TRKS; _____ SECS (10)

PART 3
Performing System Generation

The final part of this section provides directions on running the system generation program RTGEN, to configure the system planned in Parts 1 and 2. The operation of the generator is presented, followed by a step-by-step discussion of the sample generation listing in Appendix C, and details of starting-up the newly configured system. RTGEN error messages are explained at the end of part 3.

It is assumed that the user has planned his configuration and responses to generator questions with the aid of parts 1 and 2 of this section. Most of the answers required during generation will be taken directly from the worksheets.

COMPUTER CONFIGURATION

RTGEN will run on the same minimum configuration as required for an RTE-III system. The system disc controller must have the same select code during generation to be used in the RTE-III system being generated. (The Dynamic Mapping System and instructions unique to 21MX series computers are not required during generation).

GENERATOR FEATURES

SWITCH REGISTER OPTIONS

Several generation options (see Table 6-7) are selected by setting bits in the switch register. The switch register should be set before starting RTGEN and checked when restarting after an error. The switch register setting may be altered during generation at any time to reflect the user's changing requirements. This ability provides, for example, control over the amount of information listed during program relocation. Refer to the heading, "Sample Generation," for details on switch register usage during the program input phase.

HALTS

The generator normally halts in several places with the code 102077 in the memory data register. The message, "HALT 77 – SET SWR AND PRESS RUN", is displayed on the list device to notify the operator of the halt and the opportunity to change switch register (SWR) settings. (Several error halts are possible and are discussed at the end of this part).

CURRENT PAGE LINKING

Bit 14 of the switch register enables the current page linking option during module relocation. With this bit set, as each module is loaded it is checked to see if it crosses a page boundary. If it does, a linkage buffer is created before and after the module. Instructions which reference locations outside their own page are linked indirectly through an address in one of these buffers. If no buffer exists on the referencing page, if the buffer overflows, or if the link is to an external entry point, base page links are used.

The current page linking mechanism is desirable when base page space is critical. The program size is increased while the base page requirement is decreased. It might be desireable, for example, to enable this option only while the system and library modules are being loaded, since these base page links reduce the amount available to disc and memory resident programs.

NOTE

> EXT references and Assembly Language type 3 or 5 programs still use base page exclusively when establishing links. However, this is not true for subroutines referenced by these programs.

RESPONSES AND COMMENTS

Normal responses are entered as a line, left-justified, and followed by a carriage-return, line feed (CR, LF) sequence. A response in error is corrected by entering a "rubout" followed by CR,LF then retyping the response. Whenever a response is expected, one or more comments may be entered followed by the response line. A comment begins with an asterisk (*) and ends with CR,LF. A comment may also follow a response on the same line, with at least one blank separating the response and the comment. Comments are useful for documentation purposes and when switching response input from console to paper tape as discussed below, under "ANSWER TAPE."

ERROR MESSAGES

The generator produces numbered error messages of the form, "ERR XX," where XX is a two digit decimal number. These messages are presented along with their meanings and the appropriate action at the end of part 3. Note that after an error, the next response is read from the operator's console, even if the previous responses had been through paper tape.

NUMBER SYSTEMS

The generator uses octal (base 8) numbers when listing word addresses (including interrupt trap cell locations and device select codes). Responses specifying addresses must be made in octal. All other quantities, including page addresses, are expressed in decimal.

ANSWER TAPE

If desired, a paper tape may be prepared consisting of all (or some of) the responses to be entered during generation. Each response should be left-justified in a line which ends with carriage-return, line-feed (CR,LF). By selecting the appropriate switch register option, the paper tape is read rather than the console keyboard when a response is required.

If relocatable programs are to be entered from paper tape, leave a gap (a few inches of null characters) for visual identification just before starting the Parameter Phase responses. When the generator halts after the Initialization Phase, remove the answer-tape and proceed with the program Input Phase. After reading the last relocatable tape, replace the answer tape in the photoreader and position the gap under the read head after reading the final program tape.

When RTGEN halts (for example, at the start of the Partition Definition Phase), the operator has a convenient point at which to change from paper tape back to console responses and proceed interactively.

If a particular response is not known in advance when punching the answer tape, and a halt is not conveniently near, punch a response with a known error on the tape at the spot you wish to stop. When the error is encountered, control automatically reverts to the keyboard. The operator can then enter the correct response before continuing with the answer-tape. If several responses are to be entered interactively, turn off the answer-tape option in the switch register before entering the corrected response; further responses will then be read from the keyboard. To revert back to the tape, wait until a response on tape is needed, then set the answer tape bit in the switch register (position the tape if necessary), and enter a "comment" response (**comment*) through the console.

RESTARTING

If an error is discovered by the user after a response has been entered, or if RTGEN reports an uncorrectable error, it may be possible to restart the generator at some earlier point without reloading the program. Restart is accomplished by setting the program address to octal location 100 and pressing RUN. RTGEN will resume processing at the beginning of the most recently passed of 4 possible phases: either Initialization, Program Input, Parameter, or Partition Definition. Restarts after the final "/E" response at the end of the Partition Definition Phase are not allowed and will result in an irrecoverable error message. This is because RTGEN is involved in clean-up operations which cannot be repeated without reinitializing.

To restart the generator follow these steps:

a. Halt the computer and set the program address to resume at octal 100.
b. Check switch register options and make sure it applys to the restart point.
c. Position the answer-tape (if used).
d. Press Run
e. If using an answer-tape and restarting after an error message, it may be necessary to type a comment ("*") at the console to get the tape started.

GENERATOR INPUT/OUTPUT

With the exception of the disc, I/O operations to peripheral devices are carried out through SIO drivers. The System Input/Output drivers must be configured as described in the SIO System Configuration Manual. The following devices require SIO drivers if used:

Teletype	– required.
Line printer	– optional. If used, all teletype output goes to the line printer.
Photoreader	– optional. For program/response input.
Punch	– optional. For punching bootstrap and/or echoing questions and responses.
Mag Tape	– optional. For program input.

The configured SIO drivers may be punched as a single absolute tape for subsequent generations.

PREPARING RELOCATABLE TAPES

Relocatable modules should be loaded during the program input phase in the order shown below. Users with a magnetic tape unit may choose to prepare a tape containing all or most relocatables before starting the generation. 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 magnetic tape.

   Memory Resident System
   I/O Drivers
   Power Fail (DVP 43)
   System Programs written by the user
   Multi-Terminal Monitor
   Memory Resident Programs
   Real-Time Disc-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
   Editor
   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.

GENERATOR START-UP

The following steps begin the generation of the RTE-III system:

a. Apply power to all equipment. Insert any removable disc cartridges to be used into the disc drive. Refer to the appropriate disc drive hardware manual and disable any protection mechanisms on the discs to be configured during the generation.

b. Refer to the 21MX Computer Reference Manual, HP Part No. 02108-90002 for the computer under "cold start" and load the appropriate RTGEN tape through the photoreader; 92060-16029 for HP 7900, 92060-16032 for HP 7905. Similarly, load the configured SIO drivers (or load and configure individually). Note that if a magnetic tape SIO driver is used, it must be loaded last (after the generator and other SIO drivers).

c. Study Table 6-7 and set the switch register for any desired initial options. Most options can be preset at the beginning even though not used until later. Note that during the Program Input Phase, bit 15 may need to be changed if an error occurs, and bits 0 and 1 will require attention at least once in that phase.

d. Place the answer tape (if used) in the photoreader.

e. Set the starting address of the program to octal 100 and press RUN to start.

When RUN is pushed RTGEN begins asking questions on the input device. After each question is printed, the operator responds with the required answer as previously described.

SAMPLE GENERATION

The following pages discuss an actual RTE-III generation in a step-by-step procedure where the system is configured on an HP 7900 Moving Head Disc. Note that the first two questions would be different if the disc were a 7905. Refer to Part 2 steps 1 and 2 for the difference.

The listing, or generator printout, is included in this manual as Appendix C.

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 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.

```
  0?
203,0
  1?
203,0
  2?
203,0
  3?
203,0
  4?
```

Table 6-7. 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 any time)
14	Establishes current page-linking mode.	Before answering FWA BP LINKAGE? (change any time)
13	Print base page linkage listing.	Before answering FWA BP LINKAGE? (change any time)
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/during generation
4	Questions and answers are punched on the punch device.	Beginning/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/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 and 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. A comment must be prefaced with an asterisk. Comments may appear at the end of any input line and may be the only thing on a line.
3. The switch setting is examined each time the action it controls is started. Thus the setting may be examined at any time.

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. YES means 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 had been 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 would have been:

128 WORD SECTORS/TRACK?

If the answer to the AUX DISC? question was NO, then RTGEN skips to the START SCRATCH? question. Since the answer to the AUX DISC? question was YES, 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.

0

RTGEN asks if the common area should be included in the System Map for access by privileged drivers. Answer YES or NO.

PRIV. DRIVERS ACCESS COMMON?
NO

RTGEN asks if any real-time disc resident program is allowed to be locked into memory. Answer YES or NO.

RT CORE LOCK?
YES

RTGEN asks if any background disc resident program is allowed to be locked into memory. 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 size of physical memory in pages.

MEM SIZE?

Operator responds with the decimal number of pages in the system.

64

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

PRGM INPT?

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

MT

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

LIBR INPT?

Operator responds with PT, TY, or MT.

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. Even though TY is specified, parameters will be read from an answer tape if the switch register is set appropriately.

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

If there are any bad tracks, they will be reported here. Refer to Part 2 step 12.

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-III 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 first track of the system subchannel, amount of CPU space, and LU of the disc. RTGEN keeps repeating the question until NO is entered, then executes a HALT 102077. 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 and 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.

Memory Resident System
I/O Drivers
Power Fail (DVP 43)
System Programs written by the user
Multi-Terminal Monitor
Memory Resident Programs
Real-Time Disc 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
Editor
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 load-

ing 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, set 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 primary 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. In the example (see Appendix C) the operator responds with:

```
PARAMETERS
D.RTR,1,1
WHZAT,2,1
ASMB,3,95
XREF,3,96
EDITR,3,50
/E
```

RTGEN requests if there are any entry records to be changed. The numbers are assumed octal unless followed by a "D" which signifies decimal. Note the comments prefaced by the asterisk.

CHANGE ENTS?

```
*EAU MACRO'S
.MPY,RP,100200
.DIV,RP,100400
.DLD,RP,104200
.DST,RP,104400
*HFP MACROS
.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
*21MX EXTENSION MACRO'S
.MVW,RP,105777
*FFP MX
.PWR2,RP,105225
.FLUN,RP,105226
.PACK,RP,105230
.XFER,RP,105220
.XPAK,RP,105206
.XCOM,RP,105215
.DCM,RP,105216
DDINT,RP,105217
XADD,RP,105207
XSUB,RP,105210
XMPY,RP,105211
XDIV,RP,105212
/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

RTGEN requests the maximum decimal number of partitions to be defined so that table space (6 words each) may be set aside.

MAX NUMBER OF PARTITIONS?
10

RTGEN requests the first word of available main memory in base page. Operator responds with the first available octal select code number after the last I/O card.

NOTE

The generator will begin system loading following this response. Any desired switch register options affecting relocation should be set before answering. (These may be set at the beginning of generation.)

```
FWA BP LINKAGE?
 100
```

SYSTEM LOADING PHASE

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?
 40,DVA13,D

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,

EQT 10?
 27,DVR63

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 number 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 0 through 6 are predefined in the system as:

0	–	bit bucket (no action allowed)
1	–	system console
2	–	system mass storage
3	–	auxiliary mass storage
4	–	standard output device
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                      *PHOTOREADER

 6 = EQT #?
 6                      *LINE PRINTER

 7 = EQT #?
 7                      *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                     *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                     *TERMINAL
```

```
21 = EQT #?
 13                *TERMINAL
22 = EQT #?
 14                *TERMINAL
23 = EQT #?
 15                *TERMINAL
24 = EQT #?
  16               *TERMINAL
25 = EQT #?
 17                *TERMINAL
26 = EQT #?
 18                *TERMINAL
27 = EQT #?
 19                *TERMINAL
28 = EQT #?
 0
29 = EQT #?
 0
30 = EQT #?
 9                 *H.P. 2313B SUBSYSTEM
31 = EQT #?
 10                *H.P. 6940 SUBSYSTEM
32 = EQT #?
 3                 * H.P. 91200 TV INTFC
33 =EQT #?
 0
34 = EQT #?
 0
35 = EQT #?
 0
36 = EQT #?
 0
37 = EQT #?
 0
38 = EQT #?
 0
39 = EQT #?
 0
40 = EQT #?
 0
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                *SPOOL LU
52 = EQT #?
 21                *SPOOL LU
53 = EQT #?
 22                *SPOOL LU
54 = EQT #
 23                *SPOOL LU
55 = EQT #?
 24                *SPOOL LU
56 = EQT #?
 25                *SPOOL LU
57 = EQT #?
 26                *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).

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,$POWR
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
40,EQT,3
72,EQT,20
73,EQT,21
74,EQT,22
75,EQT,23
76,EQT,24
77,EQT,25
/E
```

PROGRAM LOADING PHASE

The switch register setting affects current page linking for modules relocated in this phase and determines what information will be reported by the generator.

The Memory Resident Library and the Subsystem Global Area (if any) are loaded first. SSGA is considered part of common for mapping purposes, and the remaining two common subareas are established next.

RTGEN reports the default size of Real-time common (decimal) and asks for an override. 200 (decimal) words are requested in the sample. Enter 0 for no change.

```
RT COMMON 00000
     CHANGE RT COMMON?
200
```

The starting address (octal) of Real-time common is reported:

```
RT COM 40755
```

The same sequence is followed for Background common. In the sample 200 decimal words are requested. Enter 0 for no change.

```
BG COMMON 00000
CHANGE BG COMMON?
200
```

The starting address of BG common is reported:

```
BG COM 41265
```

The last address in common is reported and the user is asked if it should be aligned at the next page boundary (alignment protects the upper neighbor, memory resident programs).

```
LWA BG COMMON 41574
ALIGN AT NEXT PAGE?
YES
```

Since YES was entered, RTGEN reports the new last address of common.

```
LWA BG COMMON 41777
```

Program loading (relocation) now continues with memory resident, real-time disc resident, and background disc resident programs. When finished, the computer will HALT.

PARTITION DEFINITION PHASE

Before beginning this phase by pushing RUN, check the switch register options (see Table 6-7) for a possible change (e.g. change from answer-tape to manual input through the console). The partition size requirements for Real-time and Background programs are listed.

```
RT PARTITION REQMTS:
  SMP     04  PAGES
  JOB     04  PAGES
  WHZAT   02  PAGES
  AUTOR   04  PAGES
```

```
BG PARTITION REQMTS:
   FMGR   06  PAGES
   EDITR  05  PAGES
   GASP   07  PAGES
   ASMB   06  PAGES
   XREF   05  PAGES
   LOADR  06  PAGES
```

The largest partition size which can be addressed by any program are reported (including base page).

```
LARGEST ADDRESSABLE PARTITION:

W/O  COM 16 PAGES
W/   COM 16 PAGES
```

The size and boundaries of System Available Memory (SAM) are established in the following sequence. SAM normally begins after the Memory Resident Program area. RTGEN reports the last word of the Memory Resident Program area and then asks if you want to align it at the next page boundary. If YES is answered, the beginning of SAM is aligned at the page boundary, not the end of the Memory Resident Program Area. That area in between the two is lost. if NO is answered, the beginning of SAM is aligned at the end of the Memory Resident Program Area and extends only to the next page boundary. Aligning the start of SAM at a page boundary affords SAM protection from unauthorized access.

```
LWA MEM RESIDENT PROG AREA  45434

ALIGN AT NEXT PAGE?
YES
```

Since YES was entered, RTGEN reports the new last address of the Memory Resident Program Area.

```
LWA MEM RESIDENT PROG AREA  45777
```

The default size of SAM is reported (decimal). Since the beginning of SAM was aligned at a page boundary above, its default size is one page.

```
SYS AV MEM:  01024 WORDS
```

If the beginning of SAM had not been aligned at a page boundary, then the area for SAM would have been the number of words between LWA MEM RESIDENT PROG AREA and the next page address.

The first page available for disc partitions ends System Available Memory. Refer to Figure 6-1. In the sample, this page number is increased by one page to add 1024 words to the default SAM size:

```
1ST DSK PG 00020

CHANGE 1ST DSK PG?
21
```

The final size of SAM is reported as two pages. This is one page (default) plus the page just added.

```
SYS AV MEM:  02048 WORDS
```

The decimal number of pages remaining for disc partitions is reported and the operator divides this area into partitions:

```
PAGES REMAINING:  00043

DEFINE PARTITIONS

1,4,RT
2,4,RT
3,14,BG
4,7,BG
5,7,BG
6,7,BG
```

RTGEN next allows the operator to override the page requirements for those programs needing dynamic space for symbol tables or buffers:

```
MODIFY PROGRAM PAGE REQUIREMENTS?

EDITR,7
ASMB,14
XREF,14
LOADR,14
/E
```

The final entries assign specific programs to be run in specific partitions. No assignments wre made in the sample.

```
ASSIGN PROGRAM PARTITIONS?
/E
```

The generation ends as the system size on disc is reported (size includes the system, and disc resident programs in memory-image format plus the relocatable library).

```
SYSTEM STORED ON DISC

SYS SIZE:  20 TRKS, 031 SECS (10)
```

RTGEN halts. (The standard halt message is printed even though the program does not continue from this point.)

INITIATING AN RTE-III SYSTEM

The newly configured system is started up in the following steps:

a. Turn-on any disc protection features which were disabled for generation.

b. Refer to 21MX the Computer Reference Manual HP Part No. 02108-90002 for program loading ("cold start") instructions. If the computer contains a build-in (e.g., ROM) disc loading program, execute it. If the disc boot is not built-in, use the paper tape bootstrap program to load the RTGEN boot punched during generation. Execute the RTGEN boot starting at location 100 (octal).

In either case, when the computer halts after the disc boot is executed, press run.

c. The message "SET TIME" should appear on the operator console.

d. File Manager initilization errors may be printed. For details on how to initialize the File Manager system, see *Batch-Spool Monitor Reference Manual* (92060-90013).

e. The operator either sets the clock to current time using the TM operator request, or enters any other request (the system starts with time set to 8:00 on approximate release date).

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	Check tape punch.
102077	Normal halt. Additional data required.	Set switch register and push RUN.

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, 14, or 30 module illegally calling a module that is not type 0, 6, 14, or 30.

Action: Revise the calling module.

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

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 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.

Action: Redefine track area or sectors per track answers.

ERR 44

Meaning: Invalid partition number entered.

Action: Reenter partition description with valid decimal number, between 1 and maximum defined during initialization.

ERR 45

Meaning: Invalid partition size.

Action: Reenter partition description with valid decimal size, between 1 and 1024 pages.

ERR 46

Meaning: Invalid partition type.

Action: Reenter partition description with valid type, BG or RT.

ERR 47

Meaning: Invalid reservation parameter.

Action: Reenter partition description. Fourth parameter must be "R" to reserve a partition.

ERR 48

Meaning: Invalid or unknown program name.

Action: Reenter response with corrected name or enter /E to end this sequence.

ERR 49

Meaning: Invalid partition number.

Action: Reenter response with corrected number or enter /E to end this sequence.

ERR 50

Meaning: Program specified is too large for partition assigned.

Action: Assign program to a larger partition or continue without assigning this program.

ERR 51

Meaning: Invalid page size. Either smaller than the program size, or larger than maximum addressable partition size.

Action: Reenter response with valid size or continue without overriding this program's page requirements.

ERR 52

Meaning: Module being relocated references an SSGA entry point but does not have proper program type to allow SSGA access.

Action: Restart generation from Parameter Phase and change the main program involved to a type allowing SSGA access or to a type 8 to delete it from the generation.

ERR 53

Meaning: The sum of all partition sizes does not equal the number of pages remaining after System Available Memory.

Action: Redefine all partitions.

ERR 54

Meaning: A subroutine or segment has declared more common than the associated main program.

Action: Recompile the main program declaring the maximum common needed by any segment or subroutine to be used.

APPENDIX A
SYSTEM COMMUNICATION AREA AND SYSTEM TABLES

APPENDIX A

This appendix contains the following information:

SYSTEM COMMUNICATION AREA

Base page locations of area used for system communication.

PROGRAM ID SEGMENT MAP

Format of ID segments kept in system area for user programs.

EQUIPMENT TABLE

Format of Equipment Table entries for RTE devices.

DEVICE REFERENCE TABLE

Format of table relating logical units to devices in Equipment Table.

DISC LAYOUT

Allocation of disc space for RTE system.

SYSTEM COMMUNICATION AREA

A block of storage in base page, starting at octal location 1647, contains the system communication area and is used by RTE-III to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, etc. The Real-Time Assembler allows relocatable programs to reference this area by absolute addresses 1647-1777 octal. 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		
01647	XI	Address of index register save area
01650	EQTA	FWA of equipment table
01651	EQT#	No. of EQT entries
01652	DRT	FWA of device reference word 1 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 9 parameters)
01701	RQP2	
01702	RQP3	
01703	RQP4	
01704	RQP5	
01705	RQP6	
01706	RQP7	
01707	RQP8	
01710	RQP9	

Octal Location		Contents	Description
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' register 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 interface card
01740		IDSDA	Disc addr. of first ID segment
01741		IDSDP	Position within sector
DEFINITION OF MEMORY ALLOCATION BASES			
01742		BPA1	FWA user base page link area
01743		BPA2	LWA user base page link area
01744		BPA3	FWA user base page link
01745		LBORG	FWA of resident library area
01746		RTORG	FWA of real-time common
01747		RTCOM	Length of real common
01750	D	RTDRA	FWA of real-time partition
01751	D	AVMEM	LWA+1 memory real-time partition
01752		BGORG	FWA of background common
01753		BGCOM	Length of background common
01754	D	BGDRA	FWA of background partition

Octal Location		Contents	Description
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 RTE Library
01764		DSCUN	# of RTE Library routines
01765		LGOTK	LGO: LU#, starting track, # of tracks (same format as ID seg word 28)
01766		LGOC	Current LGO track/sector address (same format as ID seg word 27)
01767		SFCUN	LS: LU# and disc address (same format as ID seg word 27)
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	D	FENCE	memory protect fence address
01777		BGLWA	LWA memory background partition

The letter D indicates the contents of the location are set dynamically by the dispatcher.

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.

Short ID segments requiring nine words 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 28 word one will be used.

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	X-REGISTER AT SUSPENSION (MEMORY RESIDENT PROGRAMS ONLY)															
0	Y-REGISTER AT SUSPENSION (MEMORY RESIDENT PROGRAMS ONLY)															
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	RP	# OF PAGES						MPFI			*	PARTITION #				
● 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 abort 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.
RP = Reserved partition—only for programs that request it.
MPFI = Memory protect fence index.

THE EQUIPMENT TABLE

The Equipment Table (EQT) has an entry for each I/O controller recognized by RTE-III (these entries are established by the user when the RTE-III 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	I/O Request 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 the I/O controller (lower number if a multi-board interface).

AV = I/O controller 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 "DVx," 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.
- 05 sub 0 = console (or system keyboard control device).
- 05 sub 1, 05 sub 2 = mini cartridge devices.
- 10 to 17 = unit record devices.
- 10 = plotter.
- 11 = card reader.
- 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). Base page location 1652 contains the address of the DRT first word table. Base page location 1653 contains the number of LU entries (LUMAX).

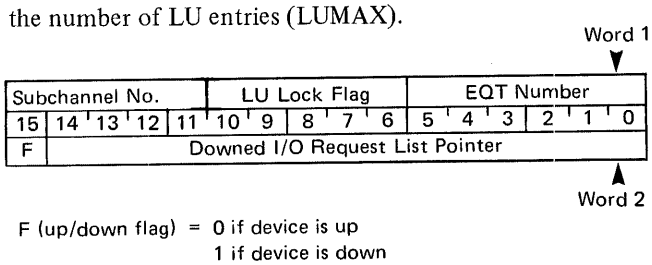


Figure A-1. Device Reference Table

The first DRT word contains the EQT entry number of the device assigned to the logical unit, and the subchannel number within the EQT entry. The second DRT word contains the logical unit's status (up or down) and a pointer to any downed I/O requests. If the pointer is less than 64, it is the LU number off of which the downed I/O requests are queued. If several LU's point to the same device, the requests are queued off the lowest LU number

(the major LU). If the pointer is greater than 64, it points to the device's downed I/O request list. There are separate tables for words one and two, with the word two table located in memory immediately following the word one table. The functions of logical units 1 through 6 are predefined in the RTE-III System as:

1 – system console
2 – reserved for system
3 – reserved for system
4 – standard output unit
5 – standard input unit
6 – standard list unit

DISC LAYOUT OF RTE-III SYSTEM

Figure A-2 diagrams the allocation of disc space by the system generator when it creates an RTE-III System. The bottom portion of the figure shows the difference between loader area in the moving head and fixed head system.

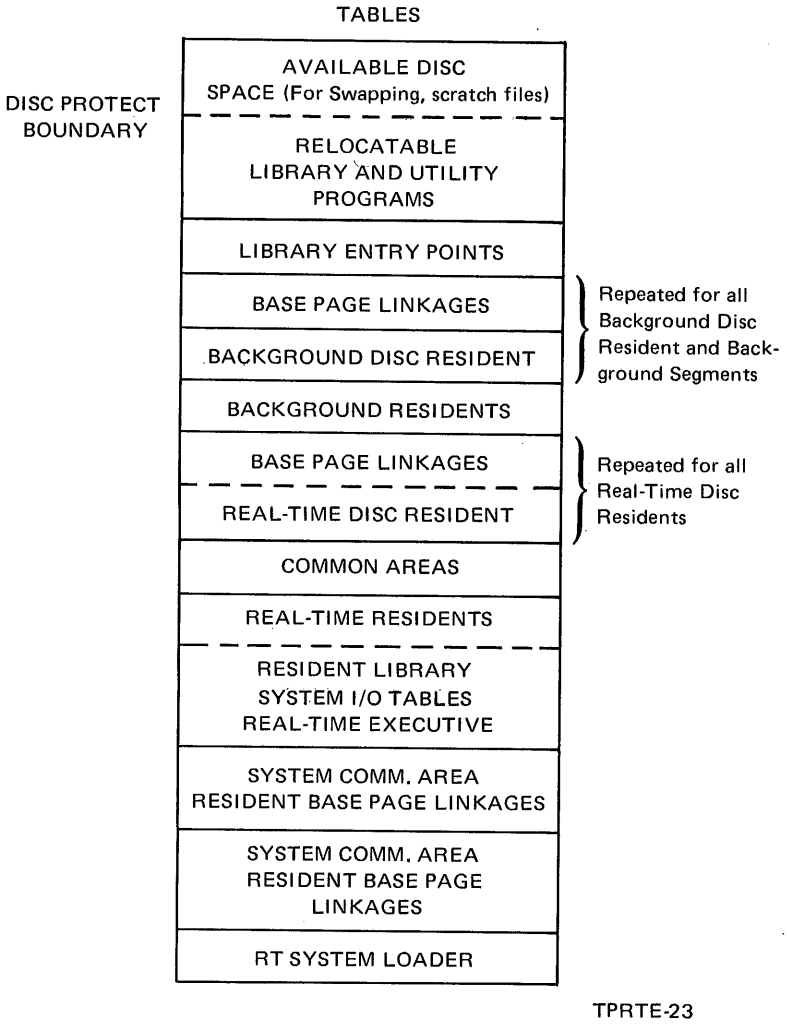


Figure A-2. Disc Space Allocation in RTE-III System

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
        ENT   $TB31
$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
        END
```

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 spliting 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.

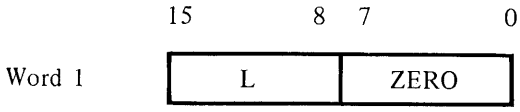
d_1. CPU B writes its updated sector back to the disc. The sector now has both updates.

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

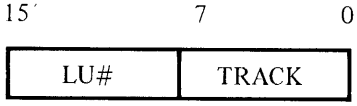


Where L is the record length in words excluding Word 1

If Word 1 = 0 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.

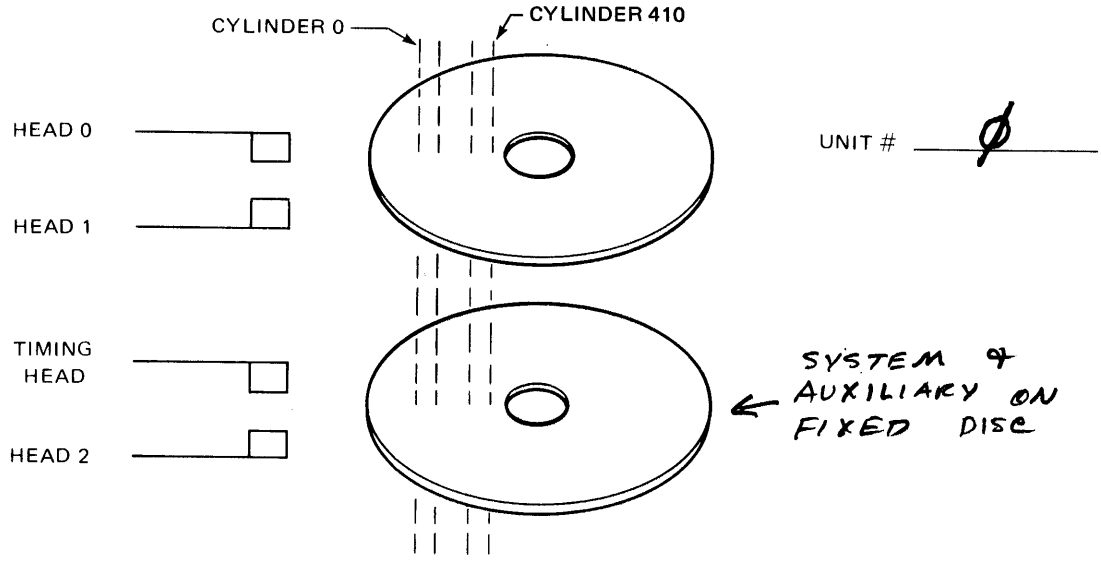
APPENDIX C
SAMPLE RTE-III
GENERATION

NOTE

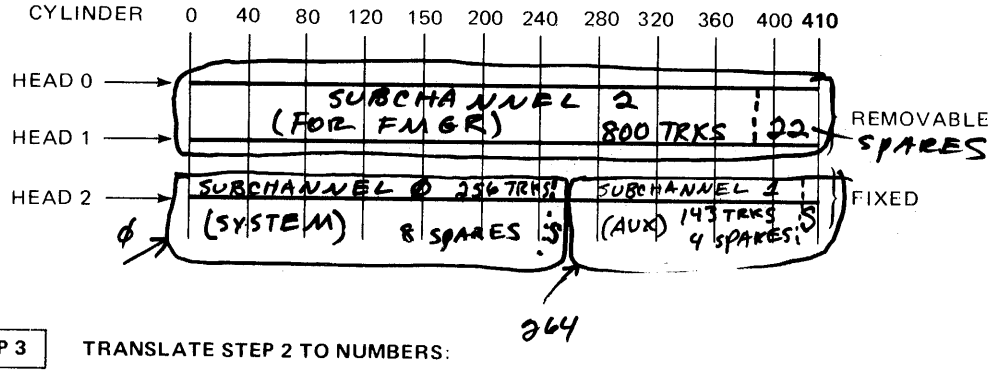
The two completed worksheets included in this Appendix are samples of an HP 7905 Disc Configuration. The generator listing is from an HP 7900 Disc Configuration

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:**

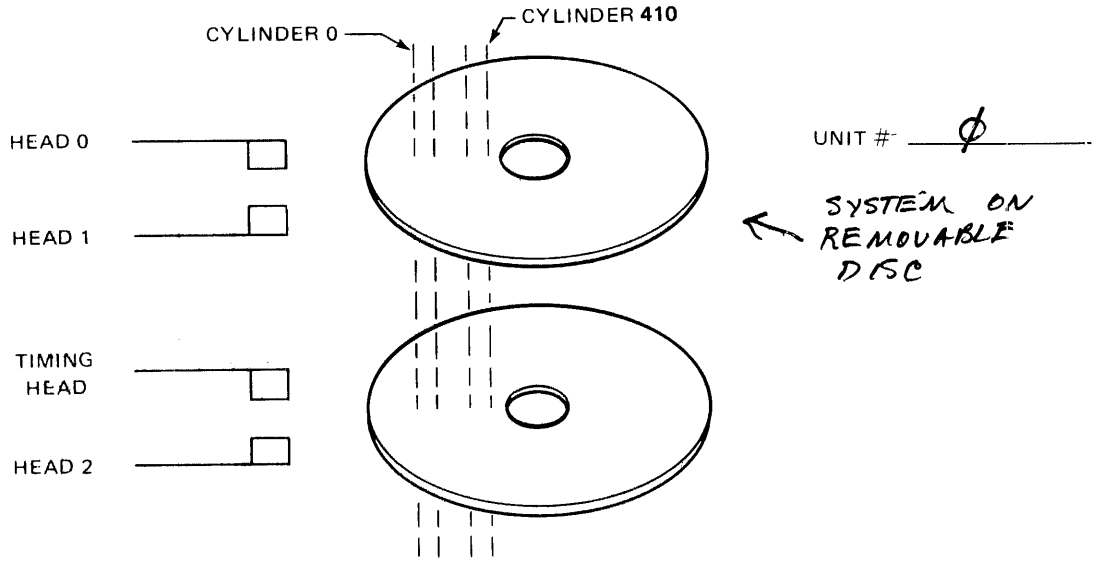


STEP 3 **TRANSLATE STEP 2 TO NUMBERS:**

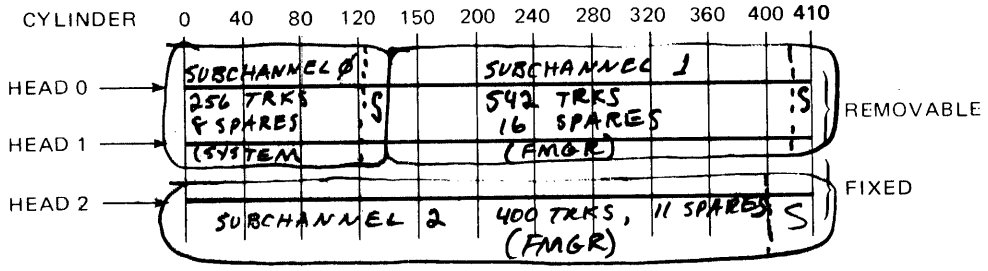
SUBCHANNEL	∅	1	2			
NUMBER OF TRACKS	256	143	8∅∅			
STARTING CYLINDER	∅	264	∅			
STARTING HEAD	2	2	∅			
NUMBER OF SURFACES	1	1	2			
NUMBER OF SPARES	8	4	22			
SYSTEM ? (✓)	✓					
AUXILIARY (✓)		✓				
SCRATCH ? (✓)						

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	∅	1	2			
NUMBER OF TRACKS	256	542	4∅∅			
STARTING CYLINDER	∅	132	∅			
STARTING HEAD	∅	∅	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?
0

PRIV. DRIVERS ACCESS COMMON?
NO

RT CORE LOCK?
YE

BG CORE LOCK?
YE

SWAP DELAY?
50
```

```
MEM SIZE?
64

PRGM INPT?
MT

LIBR INPT?
PT

PRAM INPT?
TY

INITIALIZE SUBCHNL:
  0?
NO
  2?
NO
  3?
NO
  4?
NO
  5?
NO
  6?
NO
  7?
NO

PUNCH BOOT?
YE

PUNCH BOOT?
NO


HALT 77 - SET SWR & PRESS RUN

*EOT

HALT 77 - SET SWR & PRESS RUN


 NO UNDEF EXTS


HALT 77 - SET SWR & PRESS RUN


PARAMETERS

D.RTR,1,1
WHZAT,2,1
ASMB,3,95
XREF,3,96
EDITR,3,50
/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
*21MX EXTENSION MACRO'S
* .MVW,RP,105777
*FFP MX
.PWR2,RP,105225
.FLUN,RP,105226
.PACK,RP,105230
.XFER,RP,105220
.XPAK,RP,105206
.XCOM,RP,105215
..DCM,RP,105216
DDINT,RP,105217
XADD ,RP,105207
XSUB ,RP,105210
XMPY ,RP,105211
XDIV ,RP,105212
/E


# OF BLANK ID SEGMENTS?
10

# OF BLANK BG SEG. ID SEGMENTS?
25

MAX NUMBER OF PARTITIONS?
10

FWA BP LINKAGE?
100


SYSTEM

SCHED(0099)02035 05305  92060-16020 REV.A 750505
   *$LIST  02077
   *$MESS  02435
   *$CVT3  04357
   *$CVT1  04423
   *$ABRT  04515
   *$TYPE  04427
   *$MPT1  04557
   *$MPT2  04732
   *$MPT3  04745
   *$MPT4  05035
   *$MPT5  05055
   *$MPT6  05074
   *$PARS  02545
   *$STRT  03430
   *$SCD3  05013
   *$INER  04071
   *$MPT7  05126
```

```
    *$ASTM   02500
    *$MPT8   05252
    *$IDNO   05113
    *$WORK   02045
    *$WATR   04722
    *$IDSM   04143
    *$MPSA   03566
BP LINKAGE 01640

RTIOC(0099)05444 11602  92060-16016 REV.A 750505
    *$CIC    05444
    *$XSIO   07620
    *$SYMG   10721
    *$IORQ   05657
    *$IOUP   10707
    *$IODN   10656
    *$ETEQ   11033
    *$IRT    05575
    *$XCIC   05460
    *$DEVT   10577
    *$GTIO   06676
    *$UPIO   10710
    *$CVEQ   11011
    *$YCIC   05461
    *$BLLO   01637
    *$BLUP   01640
    *$DVM    07232
    *$RSM    07411
    *$IOCL   11072
    *$LUPR   11207
    *$EQST   11352
    *$CHTO   11440
BP LINKAGE 01531

EXECD(0099)11646 13610  92060-16018 REV.A 750505
    *EXEC    11646
    *$ERMG   13443
    *$RQST   11650
    *$QTRL   13200
    *$LIBR   12062
    *$LIBX   12640
    *$DREQ   13232
    *$DREL   13316
    *$SQRL   13205
    *$SQSK   13334
    *$ERAB   13173
    *$PVCN   12202
    *$REIO   12356
    *$CREL   13074
    *$RSRE   12440
    *$ABRE   12544
    *$PWR5   11735
    *$MVBF   12437
BP LINKAGE 01476

$ALC (0099)13624 14051  92060-16017 REV.A 750505
    *$ALC    13624
    *$RTN    13725
BP LINKAGE 01465

RTIME(0099)14054 14625  92060-16014 750305
```

```
   *$TADD  14525
   *$CLCK  14054
   *$TREM  14547
   *$TIME  14256
   *$TIMV  14324
   *$ETTM  14472
   *$TIMR  14420
   *$ONTM  14371
   *$TMRQ  14575
   *$SCLK  14275
   *$BATM  14253
BP LINKAGE 01452

DISPM(0099)14647 17515  92060-16013 REV.A 750505
   *$RENT  15216
   *$BRED  17322
   *$ZZZZ  17403
   *$XEQ   14724
   *$MRMP  15144
   *$ENDS  15145
   *$MATA  15146
   *$MPFT  15147
   *$BGFR  15150
   *$RTFR  15152
   *$ALDM  16254
   *$DMAL  16257
   *$SMAP  15320
   *$PRCN  16303
   *$EMRP  15111
   *$LPSA  15112
   *$XDMP  15413
BP LINKAGE 01434

$TRRN(0099)17632 17775  92060-16019 750326
   *$TRRN  17632
   *$CGRN  17676
   *$ULLU  17720
BP LINKAGE 01432

$ASCM(0099)17776 20055  92060-16015 741120
   *$OPER  20026
   *$ERIN  20046
   *$NOPG  20036
   *$ILST  17776
   *$NOLG  20007
   *$LGBS  20017
BP LINKAGE 01424

$YSLB(0099)20056 20055  92001-16005 REV.B 741120
BP LINKAGE 01424

$BALB(0099)20056 20055  92002-16006 REV.C 750312
BP LINKAGE 01424

$SPOL(0099)20056 20055  92002-16001 REV.C 750416
BP LINKAGE 01424

DVS43(0099)20126 22430  92060-16009 REV.A 750505
   *IS43   20126
   *CS43   21576
   *$MPID  20434
```

```
   *N,SEQ  21735
BP LINKAGE 01422

DVP43(0099)22431 23211  92060-16001 REV.A  750505
   *$POWR  22431
   *IP43   23157
   *CP43   23060
BP LINKAGE 01416

DVR00(0099)23241 24320
   *I,00   23241
   *C,00   23610
   *I,01   23241
   *C,01   23610
   *I,02   23241
   *C,02   23610
BP LINKAGE 01416

DVR11(0099)24342 25471
   *C,11   25124
   *I,11   24342
BP LINKAGE 01416

DVR12(0099)25476 26025
   *I,12   25476
   *C,12   25636
BP LINKAGE 01404

DVR23(0099)26026 26671      92202-16001  REV. A
   *I,23   26026
   *C,23   26643
BP LINKAGE 01404

DVR31(0099)26705 30075
   *I,31   27552
   *C,31   27116
BP LINKAGE 01404

DVR62(0099)30101 31275  02313-16001 REV.A 750505
   *I,62   30101
   *C,62   30377
BP LINKAGE 01401

DVR63(0099)31322 33266   09611-16005 REV.A 750401
   *I,63   31346
   *C,63   32771
BP LINKAGE 01377

DVA13(0099)33304 33452  91200-16001 REV.A 740325
   *IA13   33304
   *CA13   33432
BP LINKAGE 01377

F4D.C(0099)33453 33452
BP LINKAGE 01377

F2F.B(0099)33453 33452
BP LINKAGE 01377


*# 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?
40,DVA13,D

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

EQT 10?
27,DVR63

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               * PHOTOREADER

 6 = EQT #?
6               * LINEPRINTER

 7 = EQT #?
7               * TERMINAL

 8 = EQT #?
8,0             * MAG 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              * 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              * TERMINAL

21 = EQT #?
13              * TERMINAL

22 = EQT #?
14              * TERMINAL

23 = EQT #?
15              * TERMINAL

24 = EQT #?
16              * TERMINAL

25 = EQT #?
17              * TERMINAL

26 = EQT #?
18              * TERMINAL

27 = EQT #?
19              * TERMINAL

28 = EQT #?
0

29 = EQT #?
0
```

```
30 = EQT #?
9               * H.P. 23138 SUBSYSTEM

31 = EQT #?
10              * H.P. 6940 SUBSYSTEM

32 = EQT #?
3               * H.P. 91200 TV INTFC

33 = EQT #?
0

34 = EQT #?
0

35 = EQT #?
0

36 = EQT #?
0

37 = EQT #?
0

38 = EQT #?
0

39 = EQT #?
0

40 = EQT #?
0

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              * SPOOL LU

52 * EQT #?
21              * SPOOL LU

53 * EQT #?
22              * SPOOL LU

54 * EQT #?
23              * SPOOL LU

55 * EQT #?
24              * SPOOL LU

56 * EQT #?
25              * SPOOL LU

57 * EQT #?
26              * POWER FAIL

58 * EQT #?
/E


* INTERRUPT TABLE
4,ENT,$POWR
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
40,EQT,3
72,EQT,20
73,EQT,21
74,EQT,22
75,EQT,23
76,EQT,24
77,EQT,25
/E


BP LINKAGE 01371
```

```
LIBRARY

  EQLU      40050 40125  92001-16005 741120
  .ENTR     40126 40215
  PRTN      40216 40320  92001-16005  741120
  LURQ      40321 40645  92001-16005 741120
  $ALRN     40646 40753  92001-16005  741106


SUBSYSTEM GLOBAL MODULES

SP.CL      )40754 40754  92060-16034 REV.A 750505


RT COMMON  00000
 CHANGE RT COMMON ?
200
RT COM      40755


BG COMMON  00000
 CHANGE BG COMMON ?
200
BG COM      41265


LWA BG COMMON  41574
ALIGN AT NEXT PAGE?
YES
LWA BG COMMON  41777

MEMORY RESIDENTS

PRMPT(0010)42000 42110  92001-16003 REV.B  741216

R$PN$(0010)42111 42257  92001-16003 REV.B  741002
  MESSS     42260 42352  92001-16005 741120

D.RTR(0001)42353 44271  92002-16007  750102
  P.PAS     44272 44320  92002-16006 740801

EXTND(0010)44321 44505  92060-16010 REV.A 750505
  RMPAR     44506 44530
  GETAD     44531 44546

SPOUT(0010)44547 45425  92060-16011 REV.A 750505
  .DRCT     45426 45434  92001-16005 741120


RT DISC RESIDENTS

SMP   (0030)42002 45122  92060-16007 REV.A 750505
  RNRQ      45123 45350  92001-16005 741120
  .DRCT     45351 45357  92001-16005 741120
  OPEN      45360 45545  92002-16006 741205
  READF     45546 46273  92002-16006  740801
  REIO      46274 46376  92001-16005 741120
  CLOSE     46377 46505  92002-16006  740801
```

```
  POST      46506 46534  92002-16006  740801
  $OPEN     46535 46743  92002-16006   740801
  P.PAS     46744 46772  92002-16006 740801
  RW$UB     46773 47244  92002-16006  740801
  RWND$     47245 47355  92002-16006  740801
  R/W$      47356 47511  92002-16006  740801
  .XFER     47512 47555
  RMPAR     47556 47600
  GETAD     47601 47616

JOB  (0030)42002 43405  92002-16005 REV.C 750128
  RNRQ      43406 43633  92001-16005 741120
  .DRCT     43634 43642  92001-16005 741120
  REIO      43643 43745  92001-16005 741120
  OPEN      43746 44133  92002-16006 741205
  READF     44134 44661  92002-16006  740801
  CLOSE     44662 44770  92002-16006  740801
  POST      44771 45017  92002-16006  740801
  $OPEN     45020 45226  92002-16006   740801
  P.PAS     45227 45255  92002-16006 740801
  RW$UB     45256 45527  92002-16006  740801
  RWND$     45530 45640  92002-16006  740801
  R/W$      45641 45774  92002-16006  740801
  .XFER     45775 46040
  RMPAR     46041 46063
  GETAD     46064 46101

WHZAT(0001)42002 43700  WHZAT FOR RTE-III  750210
  TMVAL     43701 43720   92001-16005 741120

AUTOR(0001)42002 42407
  TMVAL     42410 42427   92001-16005 741120
  FMTIO     42430 43660
  FRMTR     43661 46420
  DBLE      46421 46455
  SNGL      46456 46523
  .XPAK     46524 46720
  .XCOM     46721 46771
  .XFER     46772 47035
  CLRIO     47036 47040
  .FLUN     47041 47061
  .PACK     47062 47175
  IAND      47176 47205
  PAUSE     47206 47351
  .ZRLB     47352 47412
  .OPSY     47413 47452


BG DISC RESIDENTS

FMGR (0090)42002 42754  92002-16008 REV.C 750312
  FM.CM     42755 44603
  .DRCT     44604 44612  92001-16005 741120
  IFBRK     44613 44634  92001-16005 741120
  OPEN      44635 45022  92002-16006 741205
  CLOSE     45023 45131  92002-16006  740801
  $OPEN     45132 45340  92002-16006   740801
  RWND$     45341 45451  92002-16006  740801
  R/W$      45452 45605  92002-16006  740801
  RMPAR     45606 45630
```

```
   GETAD     45631 45646

FMGR0(0099)45647 45654  92002-16008  740801
   PK,,      45655 47272
   COR,A     47273 47306  92001-16005 741120
   CR,,      47307 50412
   READF     50413 51140  92002-16006  740801
   REIO      51141 51243  92001-16005 741120
   RWNDF     51244 51325  92002-16006  740801
   NAM,,     51326 51422  92002-16006 740801
   P,PAS     51423 51451  92002-16006 740801
   RWSUB     51452 51723  92002-16006  740801
   LOCK,     51724 51773
   FM,UT     51774 53116
   CREA,     53117 53170
   CREAT     53171 53446  92002-16006  741022
   .XFER     53447 53512

FMGR1(0099)45647 45762  92302-16008  741223
   .PARS     45763 47004
   C,TAB     47005 47135  92002-16008  740801
   CA,,      47136 47330  92002-16008  741119
   REA,C     47331 47413
   EE,,      47414 47455
   TR,,      47456 47677
   MR,,      47700 50034
   SE,,      50035 50221
   IF,,      50222 50432
   AB,,      50433 50625
   DP,,      50626 50665
   INPRS     50666 50762  92001-16005  741119
   READF     50763 51510  92002-16006  740801
   REIO      51511 51613  92001-16005 741120
   POSNT     51614 52050  92002-16006  740801
   P,PAS     52051 52077  92002-16006 740801
   RWSUB     52100 52351  92002-16006  740801
   WRLG      52352 52367  92002-16006 740801
   CK,SM     52370 52503
   .XFER     52504 52547
   XWRIT     52550 53250
   .OPSY     53251 53310

FMGR2(0099)45647 45660  92002-16008  740801
   IN,IT     45661 46606
   IN,,      46607 50460
   MC,,      50461 50770
   RC,,      50771 51156
   PU,,      51157 51401
   PURGE     51402 51500  92002-16006  740801
   NAM,,     51501 51575  92002-16006 740801
   J,PUT     51576 51622  92002-16006  740801
   IPUT      51623 51643  92002-16006   740801
   FID,      51644 51763
   MSC,      51764 52020
   LOCK,     52021 52070
   FM,UT     52071 53213
   CNT,      53214 53434
   FCONT     53435 53526  92002-16006 740801
   .XFER     53527 53572

FMGR3(0099)45647 45654  92002-16008   740801
```

```
  LI..       45655 47314
  DL..       47315 50534
  READF      50535 51262   92002-16006   740801
  REIO       51263 51365   92001-16005 741120
  LOCF       51366 51553   92002-16006   740801
  P.PAS      51554 51602   92002-16006 740801
  RWSUB      51603 52054   92002-16006   740801
  MSC.       52055 52111
  FM.UT      52112 53234
  .XFER      53235 53300

FMGR4(0099)45647 45663   92002-16008    740801
  ST.DU      45664 47120
  CO..       47121 47623
  F.UTM      47624 47772
  OPMES      47773 50173
  CREAT      50174 50451   92002-16006   741022
  READF      50452 51177   92002-16006   740801
  REIO       51200 51302   92001-16005 741120
  RWNDF      51303 51364   92002-16006   740801
  LOCF       51365 51552   92002-16006   740801
  NAM..      51553 51647   92002-16006 740801
  P.PAS      51650 51676   92002-16006 740801
  RWSUB      51677 52150   92002-16006   740801
  FM.UT      52151 53273
  CREA.      53274 53345
  CK.SM      53346 53461
  .XFER      53462 53525

FMGR5(0099)45647 45656   92002-16008    740801
  ??..       45657 47772   92002-16008   740801
  RU..       47773 50433
  RP..       50434 51575
  TL..       51576 51615
  READF      51616 52343   92002-16006   740801
  REIO       52344 52446   92001-16005 741120
  LOCF       52447 52634   92002-16006   740801
  P.PAS      52635 52663   92002-16006 740801
  RWSUB      52664 53135   92002-16006   740801
  J.PUT      53136 53162   92002-16006   740801
  IPUT       53163 53203   92002-16006    740801
  ID.A       53204 53273
  BUMP.      53274 53332   92002-16006   741025
  SET.T      53333 53361   92002-16006 740801
  TL.        53362 53411   92002-16006   741025
  ST.TM      53412 53446   92002-16006 741223
  .XFER      53447 53512

FMGR6(0099)45647 45657   92002-16008    740801
  CN..       45660 45720
  JO..       45721 47016
  RNRQ       47017 47244   92001-16005 741120
  KCVT       47245 47257   92001-16005 741120
  EO..       47260 50103
  MESSS      50104 50176   92001-16005 741120
  OF..       50177 50272
  LG..       50273 50367
  NAMF       50370 50541   92002-16006   740801
  READF      50542 51267   92002-16006   740801
  REIO       51270 51372   92001-16005 741120
  POST       51373 51421   92002-16006   740801
```

```
  NAM..     51422 51516  92002-16006 740801
  P.PAS     51517 51545  92002-16006 740801
  RWSUB     51546 52017  92002-16006  740801
  SPOPN     52020 52070  92002-16006  741025
  SET.T     52071 52117  92002-16006 740801
  ST.TM     52120 52154  92002-16006 741223
  B.FLG     52155 52223  92002-16006 741118
  LULU.     52224 52317  92002-16006 740801
  RANGE     52320 52343  92002-16006 740801
  ONOFF     52344 52707  92002-16006 750128
  EX.TM     52710 53076  92002-16006 741008
  IPUT      53077 53117  92002-16006   740801
  FREE.     53120 53167  92002-16006 740801
  LU.CL     53170 53231  92002-16006  740801
  AVAIL     53232 53324  92002-16006 741231
  .XFER     53325 53370

FMGR7(0099)45647 45656  92002-16008   740801
  CL..      45657 46140
  NX.JB     46141 47006
  RNRQ      47007 47234  92001-16005 741120
  LU..      47235 50307
  KCVT      50310 50322  92001-16005 741120
  CS..      50323 50520
  READF     50521 51246  92002-16006  740801
  REIO      51247 51351  92001-16005 741120
  FSTAT     51352 51376  92002-16006  740801
  POST      51377 51425  92002-16006  740801
  P.PAS     51426 51454  92002-16006 740801
  RWSUB     51455 51726  92002-16006  740801
  SPOPN     51727 51777  92002-16006  741025
  B.FLG     52000 52046  92002-16006 741118
  LULU.     52047 52142  92002-16006 740801
  RANGE     52143 52166  92002-16006 740801
  AVAIL     52167 52261  92002-16006 741231
  .XFER     52262 52325

FMGR8(0099)45647 45655  92002-16008   740801
  SA..      45656 46630
  SP..      46631 47450
  MS..      47451 47744
  READF     47745 50472  92002-16006  740801
  REIO      50473 50575  92001-16005 741120
  RWNDF     50576 50657  92002-16006  740801
  LOCF      50660 51045  92002-16006  740801
  P.PAS     51046 51074  92002-16006 740801
  RWSUB     51075 51346  92002-16006  740801
  IPUT      51347 51367  92002-16006   740801
  CREA.     51370 51441
  CREAT     51442 51717  92002-16006  741022
  NAM..     51720 52014  92002-16006 740801
  CK.SM     52015 52130
  ID.A      52131 52220
  WRISS     52221 52257  92002-16006 740801
  READ.     52260 52304  92002-16006  740801
  .XFER     52305 52350
  %WRIS     52351 52640
  SREAD     52641 53376
  .OPSY     53377 53436

EDITR(0050)42002 46456  92002-16010 REV.C  750413
```

```
  REIO      46457 46561   92001-16005 741120
  CREAT     46562 47037   92002-16006  741022
  OPEN      47040 47225   92002-16006 741205
  READF     47226 47753   92002-16006  740801
  CLOSE     47754 50062   92002-16006  740801
  CLOSE     47754 50062   92002-16006  74080
  $OPEN     50160 50366   92002-16006   740801
  P.PAS     50367 50415   92002-16006 740801
  RW$UB     50416 50667   92002-16006  740801
  RWND$     50670 51000   92002-16006  740801
  R/W$      51001 51134   92002-16006  740801
  .XFER     51135 51200
  RMPAR     51201 51223
  GETAD     51224 51241

GASP (0080)42002 43377
  RNRQ      43400 43625   92001-16005 741120
  REIO      43626 43730   92001-16005 741120
  KCVT      43731 43743   92001-16005 741120
  PARSE     43744 43763   92001-16005 741120
  OPEN      43764 44151   92002-16006 741205
  READF     44152 44677   92002-16006  740801
  CLOSE     44700 45006   92002-16006  740801
  POST      45007 45035   92002-16006  740801
  $OPEN     45036 45244   92002-16006   740801
  P.PAS     45245 45273   92002-16006 740801
  RW$UB     45274 45545   92002-16006  740801
  RWND$     45546 45656   92002-16006  740801
  R/W$      45657 46012   92002-16006  740801
  G1CEX     46013 46114
  ST.LU     46115 46244   92002-16001 741025
  .DRCT     46245 46253   92001-16005 741120
  G1ROT     46254 46403   92002-16001  741027
  G1CDJ     46404 46770
  G1CCJ     46771 47332
  G1CDS     47333 50662   92002-16001   741030
  G1C??     50663 51473   92002-16001 741027
  G1STM     51474 51666   92002-16001  740807
  CNUMD     51667 51706   92001-16005 741120
  G0QIP     51707 52122   92002-16001  741007
  G1CKS     52123 52712
  G1CIN     52713 54174
  CREAT     54175 54452   92002-16006  741022
  NAM..     54453 54547   92002-16006 740801
  G1CDA     54550 55124
  PURGE     55125 55223   92002-16006  740801
  .XFER     55224 55267
  RMPAR     55270 55312
  GETAD     55313 55330

ASMB (0095)42002 47547   92060-16022 REV. 750420

ASMBD(0099)47550 50356   92060-16023 REV. 750420

ASMB1(0099)47550 51544   92060-16024 REV. 750420

ASMB2(0099)47550 52046   92060-16025 REV. 750420

ASMB3(0099)47550 50637   92060-16026 REV. 750420

ASMB4(0099)47550 51301   92060-16027 REV. 750420
```

```
XREF (0096)42002 50240  92060-16028 REV. 750420
  .OPSY    50241 50300

LOADR(0090)42002 53211    92060-16005 REV.A 750505


HALT 77 - SET SWR & PRESS RUN

RT PARTITION REQMTS:
  SMP   04 PAGES
  JOB   04 PAGES
  WHZAT 02 PAGES
  AUTOR 04 PAGES

BG PARTITION REQMTS:
  FMGR  06 PAGES
  EDITR 05 PAGES
  GASP  07 PAGES
  ASMB  06 PAGES
  XREF  05 PAGES
  LOADR 06 PAGES

LARGEST ADDRESSABLE PARTITION:
W/O COM 16 PAGES
W/  COM 16 PAGES


LWA MEM RESIDENT PROG AREA 45434
ALIGN AT NEXT PAGE?
YES
LWA MEM RESIDENT PROG AREA 45777

SYS AV MEM:  01024 WORDS

1ST DSK PG 00020
 CHANGE 1ST DSK PG?
21

SYS AV MEM:  02048 WORDS

PAGES REMAINING: 00043
DEFINE PARTITIONS
1,4,RT
2,4,RT
3,14,BG
4,7,BG
5,7,BG
6,7,BG
/E

MODIFY PROGRAM PAGE REQUIREMENTS?
EDITR,7
ASMB,14
XREF,14
LOADR,14
/E

ASSIGN PROGRAM PARTITIONS?
/E
```

```
SYSTEM STORED ON DISC
SYS SIZE: 20 TRKS, 031 SECS(10)

HALT 77 - SET SWR & PRESS RUN
```

APPENDIX D
SUMMARY OF EXEC CALLS

ASSEMBLY LANGUAGE FORMAT

```
        EXT  EXEC        Used to link program
                         to RTE-III.
         .
         .
         .
        JSB  EXEC        Transfer control to
                         RTE-III.

        DEF  *+n+1       Defines point of
                         return from RTE-III, 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 2,, 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
ISTA1 NOP	Word 5 of EQT entry returned here
ISTA2 NOP	Optional parameter for word 4 of EQT
ISTA3 NOP	Optional parameter for LU status

FORTRAN:

CALL EXEC(ICODE,ICNWD,ISTA1,ISTA2,ISTA3)

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 resuability 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 = 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
IBUFR  BSS      Optional buffer of n words
IBUFL  DEC      Same n; words (+) or
                characters (-)
```

FORTRAN:

```
REG = EXEC(ICODE,INAME,IPRM1 . . . IPRM5,
      IBUFR, IBUFL)
```

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
IPRM5
```

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)
```

STRING PASSAGE

Purpose:

Pick up command string or pass buffer to "Father."

Assembly:

```
ICODE  DEC 14
IRCOD  DEC      1=retrieve command string
                2=pass buffer
IBUFR  BSS      Buffer of n words
IBUFL  DEC      Same n; words (+) or
                characters (-)
```

FORTRAN:

```
CALL EXEC(ICODE,IRCOD,IBUFR,IBUFL)
```

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.
```

FORTRAN:

```
CALL EXEC(ICODE,IPROG,IRESL,MTPLE,IOFST)
```

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     0x0000=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)
```

PARTITION STATUS

Purpose:

Return status on specified partition

Assembly:

```
ICODE  DEC 25
IPART  DEC n   Partition number
IPAGE  BSS 1   Starting page number returned here
IPNUM  BSS 1   Number of pages returned here
ISTAT  BSS 1   Partition status
```

FORTRAN:

```
CALL EXEC (ICODE,IPART,IPAGE,IPNUM,ISTAT)
```

APPENDIX E
SUMMARY OF ERROR MESSAGES

OPERATOR REQUEST ERROR MESSAGES

When an operator request is in error, RTE-III 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.
CMD IGNORED-NO MEM	Not enough system available memory exists for storing the program's command string. Re-enter the command (RU, ON, GO) or enter the inhibit (IH) form of the command.

EXEC CALL ERROR MESSAGES

When RTE-III 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-III 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)
DM	Program tried to access a page not included in its logical memory (similar to MP)

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 = Not enough parameters

DR02 = Number of tracks is ⩽ 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-III 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 (not 1, 2, 3, or 4).

SC07 = Prohibited core lock attempted.

SC10 = Not enough system available memory for string passage.

ERROR CODES FOR I/O CALLS

IO00 = Illegal class number. Outside table not allocated, or bad security code

IO01 = Not enough parameters

IO02 = Illegal logical unit or less than 5 parameters and X-bit set

IO03 = Not used

IO04 = Illegal user buffer. Extends beyond FG/BG area or not enough system memory to buffer the request

IO05 = Illegal disc track or sector

IO06 = Reference to a protected track; or using LG tracks before assigning them (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 call

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, logical unit reference less than one, or logical unit not locked to caller

INPUT/OUTPUT ERROR MESSAGES

ILLEGAL INTERRUPTS

When an illegal interrupt occurs, RTE-III prints this message:

ILL INT *xx*

Where *xx* is the octal channel number.

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

EQUIPMENT ERROR MESSAGES

Message	Meaning
I/O ET L *#x* E *#y* S *#z*	End-of-tape condition on LU *#x*, defined by EQT *#y* subchannel *#z*. Correct the condition and set I/O controller (EQT) UP.
I/O TO L *#x* E *#y* S *#z*	Device (LU *#x*) defined by EQT *#y* subchannel *#z* has timed out. Examine device. Correct problem and set I/O controller (EQT) UP.
I/O NR L *#x* E *#y* S *#z*	Device (LU *#x*) defined by EQT *#y* subchannel *#z* is not ready. Make it ready and set I/O controller (EQT) UP.

I/O PE L #*x* E #*y* S #*z* — Parity error in data transmission from device (LU #*x*) defined by EQT #*y* subchannel #*z*. 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-III 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.

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 complier. 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 system will output a message to the operator:

I/O ET L #*x* E #*y* S #*z*

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

UP, #*x*

If an EOT does not cause the tape reader to be set down, the RTE-III 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-III when FTN attempts to write on the LG tracks (RTE-III aborts FTN).

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.

ALGOL ERRORS

More than one source tape can be compiled into one ALGOL program. 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-III System will output a message to the operator:

I/O ET L *#x* E *#y* S *#z*

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

UP, *#x*

If an EOT does not cause the tape reader to set down, the RTE-III 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 console 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-III when ALGOL attempts to write on the LG tracks (RTE-III aborts ALGOL).

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.

ASSEMBLER ERRORS

When a paper tape is being input through the tape reader, RTE-III 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-III system will output a message to the operator:

I/O ET L *#x* E *#y* S *#z*

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

UP, *#x*

If an EOT does not cause the tape reader to be set down, the RTE-III 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 console 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 console signals:

$END ASMB NPRG

The current assembly aborts.

RTE-III generates two messages when ASMB attempts to write on the load-and-go tracks (RTE-III aborts ASMB).

IO06

IO09

IO06 means that the LG 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 seperate line, above each error diagnostic in the listing.

RELOCATING LOADER ERRORS

Messages are printed in this format:

/LOADR: *message*

WARNING (W) MESSAGE

W17 – Number of pages required by the program exceeds the partition size. The loader cannot find a partition large enough for the program. It can be relocated successfully but cannot be executed. You can generate a new system containing a partition large enough for the program or you can revise the program.

"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 the LG track area is specified, but no program exists in that area.)

L09 – Record out of sequence

L10 – Operator request parameter error. GO requests may be retyped; ON requests may not.

L11 – Operator attempted to replace or purge a memory resident program.

L12 – LG track area used without resetting (*input option = 2* in "GO"). *Input option* was not specified as 99 previously.

L13 – LG track area has been illegally reset (i.e. overwritten). Program addition on this area not allowed if it has already been specified for program input. Or area was once used for force loading with *input option* = 99 and it is again being used with *input option* = 99, (must = 2).

L14 – Assembler or compiler 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.

L16 – Illegal partition number. Value must be in the range from 1 through 64.

L17 – Number of pages required by the program exceeds assigned partition size. A specific partition has been assigned for this program. The program requires more pages than are available in the partition.

L18 – Total number of pages required exceeds 32. The sum of required pages must be in the range from 1 through 32.

ADDITIONAL MESSAGES

NO BLANK ID SEGMENT

This message is printed when no 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,*name*,8 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

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 to scan the library again.

SYSTEM HALTS

Several system halts are located within the protected system area. They are as follows:

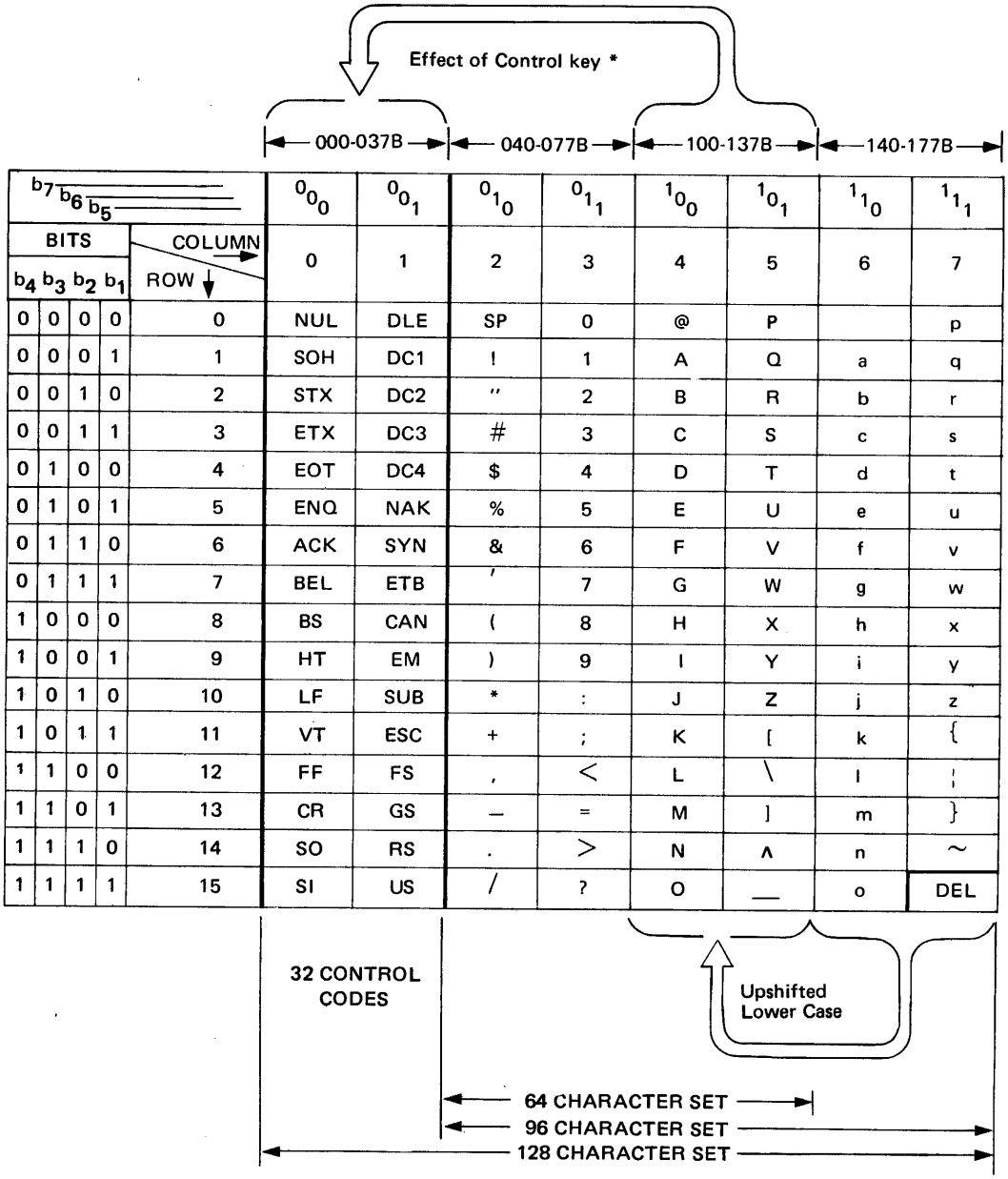
Halt	Reason
0	JSB EXC with memory protect off.
2	Tried to execute location 2.
3	Tried to execute location 3.
4	System was in halt mode when power failed; or, no EQT entry for DVP43 power fail routine.

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* / ,,*lu*	Set EQT (I/O controller) or LU (I/O device) down.
EQ,*eqt*	Print status of EQT entry *eqt.*
EQ,*eqt*[,*UN* / ,*BU*]	*UN* = delete buffering *BU* = specify buffering
GO / GOIH ,*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 / ONIH ,*name [,NOW] [, p1 [,... [,p5]]]]]*	Schedules *name. NOW* means ignore time values.
PR,*name,numb*	Set priority of *name = numb*
RU / RUIH ,*name [,p1[, . . . [,p5]]]]*	Schedule program *name*
RT,*name*	Release *name's* tracks
SS,*name*	Suspend *name*
ST,*name*	Print status of *name*

ST,0	Print name and partition number of current program
ST,*part numb*	Prints name of current program in *part numb.*
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 *eqt* and any associated LU's up.

APPENDIX G
HP CHARACTER SET



b_7 b_6 b_5 →					0_0	0_1	1_0	1_1	0_0	0_1	1_0	1_1
BITS				COLUMN →	0	1	2	3	4	5	6	7
b_4	b_3	b_2	b_1	ROW ↓								
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

EXAMPLE: The representation for the character "K" (column 4, row 11) is.

	b_7	b_6	b_5	b_4	b_3	b_2	b_1
BINARY	1	0	0	1	0	1	1
OCTAL	1		1			3	

* Depressing the Control key while typing an upper case letter produces the corresponding control code on most terminals. For example, Control-H is a backspace.

9206- 1A

HEWLETT-PACKARD CHARACTER SET FOR COMPUTER SYSTEMS

This table shows HP's implementation of ANS X3.4-1968 (USASCII) and ANS X3.32-1973. Some devices may substitute alternate characters from those shown in this chart (for example, Line Drawing Set or Scandanavian font) Consult the manual for your device.

The left and right byte columns show the octal patterns in a 16 bit word when the character occupies bits 8 to 14 (left byte) or 0 to 6 (right byte) and the rest of the bits are zero. To find the pattern of two characters in the same word, add the two values. For example, "AB" produces the octal pattern 040502. (The parity bits are zero in this chart.)

The octal values 0 through 37 and 177 are control codes. The octal values 40 through 176 are character codes.

Decimal Value	Octal Values		Mnemonic	Graphic[1]	Meaning
	Left Byte	Right Byte			
0	000000	000000	NUL	$^{N}_{U}$	Null
1	000400	000001	SOH	$^{S}_{H}$	Start of Heading
2	001000	000002	STX	$^{S}_{X}$	Start of Text
3	001400	000003	ETX	$^{E}_{X}$	End of Text
4	002000	000004	EOT	$^{E}_{T}$	End of Transmission
5	002400	000005	ENQ	$^{E}_{Q}$	Enquiry
6	003000	000006	ACK	$^{A}_{K}$	Acknowledge
7	003400	000007	BEL		Bell, Attention Signal
8	004000	000010	BS	$^{B}_{S}$	Backspace
9	004400	000011	HT	$^{H}_{T}$	Horizontal Tabulation
10	005000	000012	LF	$^{L}_{F}$	Line Feed
11	005400	000013	VT	$^{V}_{T}$	Vertical Tabulation
12	006000	000014	FF	$^{F}_{F}$	Form Feed
13	006400	000015	CR	$^{C}_{R}$	Carriage Return
14	007000	000016	SO	$^{S}_{O}$	Shift Out } Alternate
15	007400	000017	SI	$^{S}_{I}$	Shift In } Character Set
16	010000	000020	DLE	$^{D}_{L}$	Data Link Escape
17	010400	000021	DC1	D_1	Device Control 1 (X-ON)
18	011000	000022	DC2	D_2	Device Control 2 (TAPE)
19	011400	000023	DC3	D_3	Device Control 3 (X-OFF)
20	012000	000024	DC4	D_4	Device Control 4 ($\overline{\text{TAPE}}$)
21	012400	000025	NAK	$^{N}_{K}$	Negative Acknowledge
22	013000	000026	SYN	$^{S}_{Y}$	Synchronous Idle
23	013400	000027	ETB	$^{E}_{B}$	End of Transmission Block
24	014000	000030	CAN	$^{C}_{N}$	Cancel
25	014400	000031	EM	$^{E}_{M}$	End of Medium
26	015000	000032	SUB	$^{S}_{B}$	Substitute
27	015400	000033	ESC	$^{E}_{C}$	Escape[2]
28	016000	000034	FS	$^{F}_{S}$	File Separator
29	016400	000035	GS	$^{G}_{S}$	Group Separator
30	017000	000036	RS	$^{R}_{S}$	Record Separator
31	017400	000037	US	$^{U}_{S}$	Unit Separator
127	077400	000177	DEL	▩	Delete, Rubout[3]

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
32	020000	000040		Space, Blank
33	020400	000041	!	Exclamation Point
34	021000	000042	"	Quotation Mark
35	021400	000043	#	Number Sign, Pound Sign
36	022000	000044	$	Dollar Sign
37	022400	000045	%	Percent
38	023000	000046	&	Ampersand, And Sign
39	023400	000047	'	Apostrophe, Acute Accent
40	024000	000050	(	Left (opening) Parenthesis
41	024400	000051	)	Right (closing) Parenthesis
42	025000	000052	*	Asterisk, Star
43	025400	000053	+	Plus
44	026000	000054	,	Comma, Cedilla
45	026400	000055	–	Hyphen, Minus, Dash
46	027000	000056	.	Period, Decimal Point
47	027400	000057	/	Slash, Slant
48	030000	000060	0	} Digits, Numbers
49	030400	000061	1	
50	031000	000062	2	
51	031400	000063	3	
52	032000	000064	4	
53	032400	000065	5	
54	033000	000066	6	
55	033400	000067	7	
56	034000	000070	8	
57	034400	000071	9	
58	035000	000072	:	Colon
59	035400	000073	;	Semicolon
60	036000	000074	<	Less Than
61	036400	000075	=	Equals
62	037000	000076	>	Greater Than
63	037400	000077	?	Question Mark

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
64	040000	000100	@	Commercial At
65	040400	000101	A	Upper Case Alphabet, Capital Letters
66	041000	000102	B	
67	041400	000103	C	
68	042000	000104	D	
69	042400	000105	E	
70	043000	000106	F	
71	043400	000107	G	
72	044000	000110	H	
73	044400	000111	I	
74	045000	000112	J	
75	045400	000113	K	
76	046000	000114	L	
77	046400	000115	M	
78	047000	000116	N	
79	047400	000117	O	
80	050000	000120	P	
81	050400	000121	Q	
82	051000	000122	R	
83	051400	000123	S	
84	052000	000124	T	
85	052400	000125	U	
86	053000	000126	V	
87	053400	000127	W	
88	054000	000130	X	
89	054400	000131	Y	
90	055000	000132	Z	
91	055400	000133	[	Left (opening) Bracket
92	056000	000134	\	Backslash, Reverse Slant
93	056400	000135	]	Right (closing) Bracket
94	057000	000136	^ ↑	Caret, Circumflex; Up Arrow[4]
95	057400	000137	_ ←	Underline; Back Arrow[4]

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
96	060000	000140	`	Grave Accent[5]
97	060400	000141	a	Lower Case Letters[5]
98	061000	000142	b	
99	061400	000143	c	
100	062000	000144	d	
101	062400	000145	e	
102	063000	000146	f	
103	063400	000147	g	
104	064000	000150	h	
105	064400	000151	i	
106	065000	000152	j	
107	065400	000153	k	
108	066000	000154	l	
109	066400	000155	m	
110	067000	000156	n	
111	067400	000157	o	
112	070000	000160	p	
113	070400	000161	q	
114	071000	000162	r	
115	071400	000163	s	
116	072000	000164	t	
117	072400	000165	u	
118	073000	000166	v	
119	073400	000167	w	
120	074000	000170	x	
121	074400	000171	y	
122	075000	000172	z	
123	075400	000173	{	Left (opening) Brace[5]
124	076000	000174	¦	Vertical Line[5]
125	076400	000175	}	Right (closing) Brace[5]
126	077000	000176	~	Tilde, Overline[5]

9206-1C

Notes: [1]This is the standard display representation. The software and hardware in your system determine if the control code is displayed, executed, or ignored. Some devices display all control codes as "||", "@", or space.

[2]Escape is the first character of a special control sequence. For example, ESC followed by "J" clears the display on a 2640 terminal.

[3]Delete may be displayed as "__", "@", or space.

[4]Normally, the caret and underline are displayed. Some devices substitute the up arrow and back arrow.

[5]Some devices upshift lower case letters and symbols (` through ~) to the corresponding upper case character (@ through ^). For example, the left brace would be converted to a left bracket.

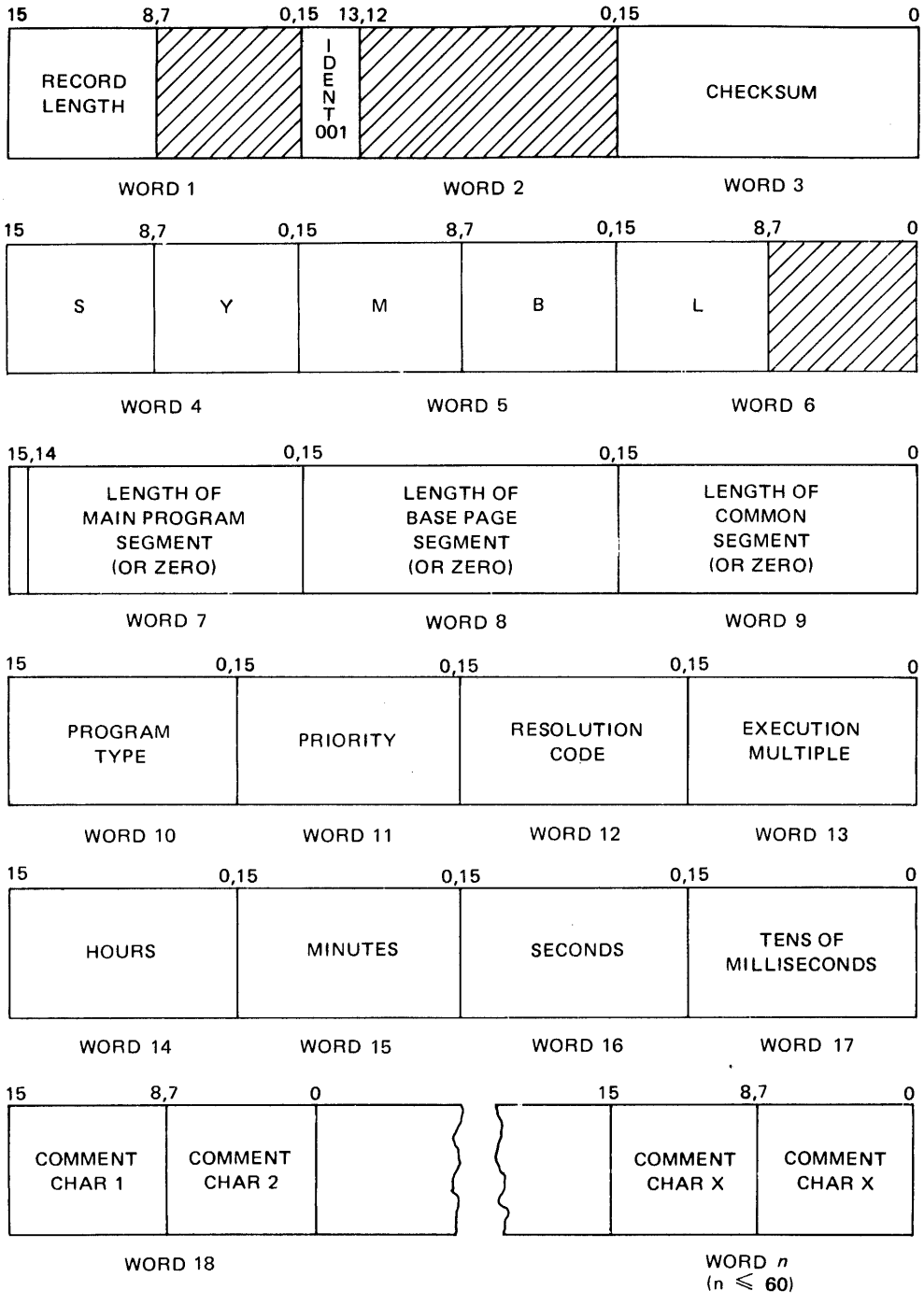
RTE SPECIAL CHARACTERS:

Mnemonic	Octal Value	Use
SOH (Control A)	1	Backspace
EM (2600 Backspace) (Control Y)	31	Backspace
BS (Control H)	10	Backspace
EOT (Control D)	4	Simulate End Tape

APPENDIX H
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.

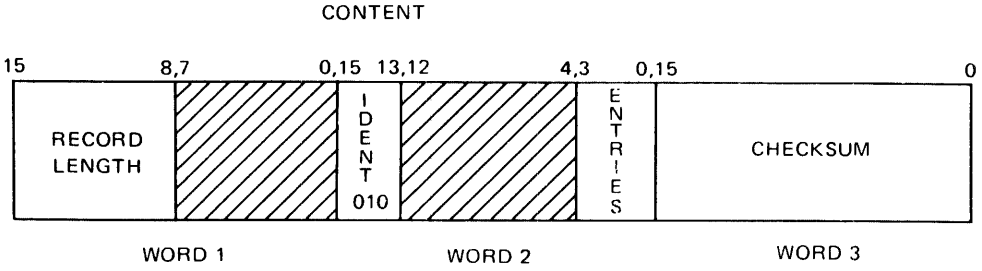
SYMBL: FIVE CHARACTER NAME OF PROGRAM

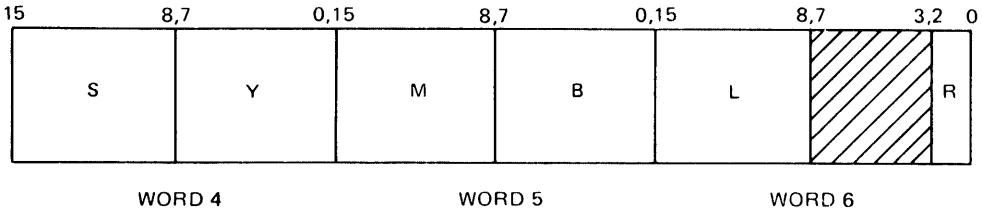
A/C: BINARY TAPE PRECESSION
= 0 IF ASSEMBLER PRODUCED
= 1 IF COMPILER PRODUCED

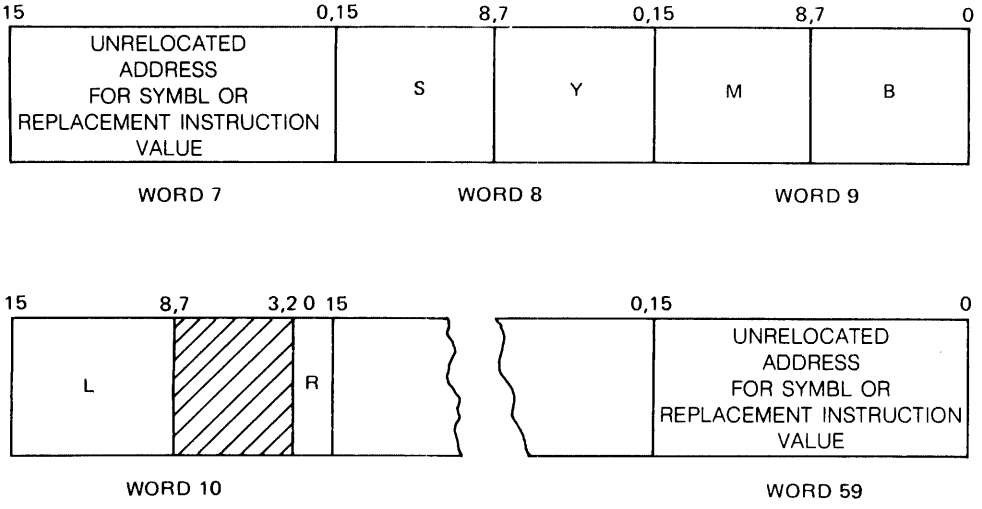
HATCH-MARKED AREAS SHOULD BE ZERO-FILLED WHEN THE RECORDS ARE GENERATED

TODS—14

ENT RECORD

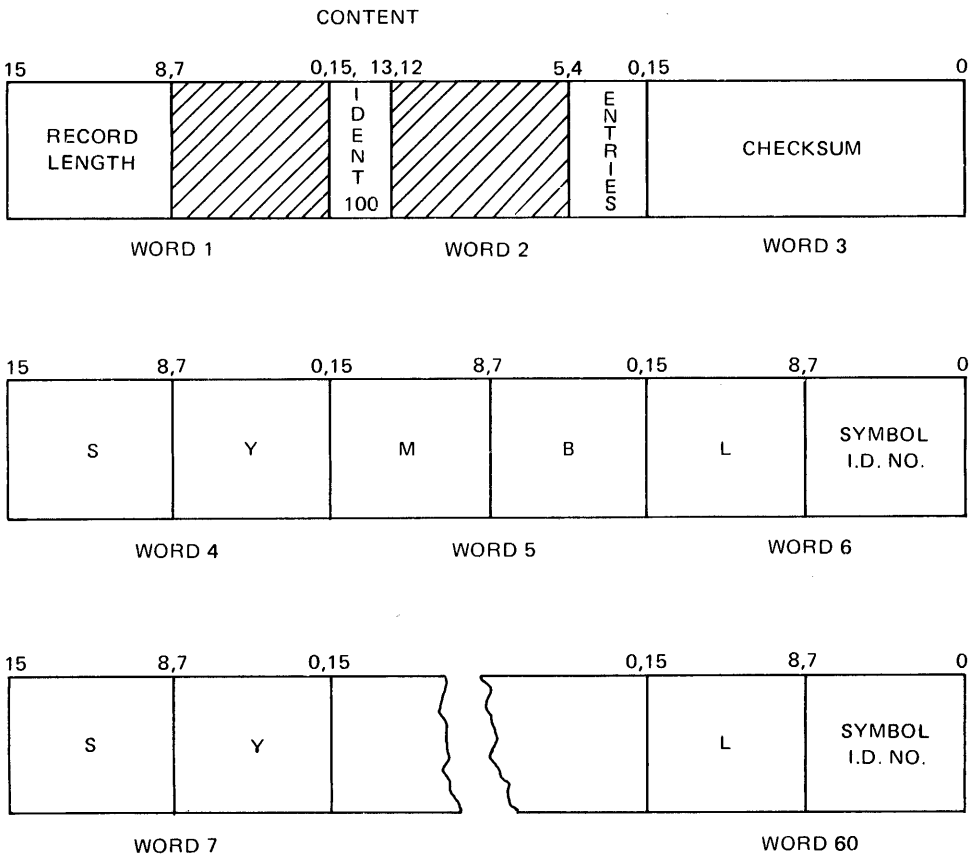






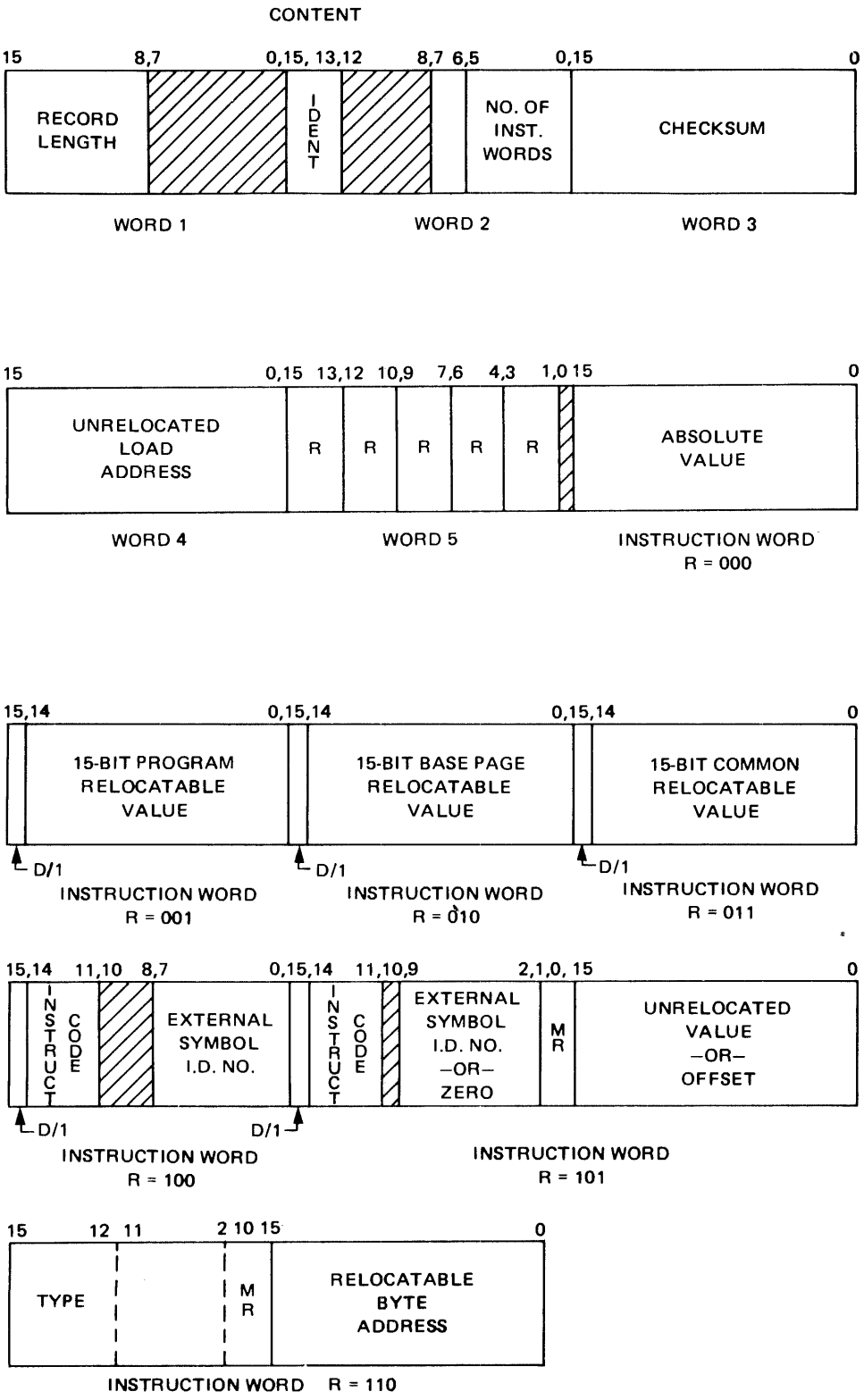
TODS—15

EXT RECORD



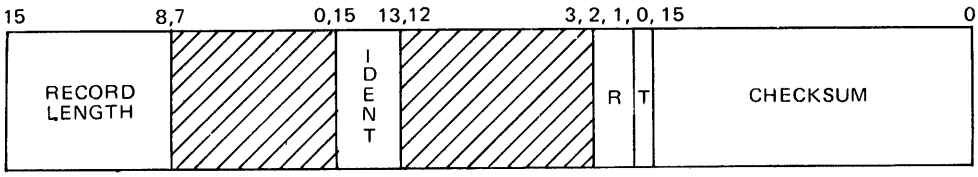
TODS—16

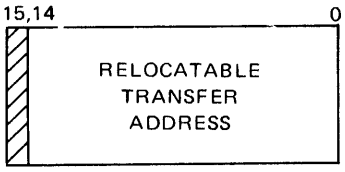
DBL RECORD


CONTENT
EXPLANATION
15
8,7
0,15, 13,12
8,7 6,5
0,15
0
RECORD LENGTH
IDENT
NO. OF INST. WORDS
CHECKSUM
WORD 1
WORD 2
WORD 3
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
15
0,15 13,12 10,9 7,6 4,3 1,0 15
0
UNRELOCATED LOAD ADDRESS
R
R
R
R
R
ABSOLUTE VALUE
WORD 4
WORD 5
INSTRUCTION WORD
R = 000
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
110 = BYTE REFERENCE
15,14
0,15,14
0,15,14
0
15-BIT PROGRAM RELOCATABLE VALUE
15-BIT BASE PAGE RELOCATABLE VALUE
15-BIT COMMON RELOCATABLE VALUE
D/1
D/1
D/1
INSTRUCTION WORD
R = 001
INSTRUCTION WORD
R = 010
INSTRUCTION WORD
R = 011
R1 IS RELOCATION INDICATOR FOR INSTRUCTION WORD1; R2, FOR INSTRUCTION WORD2; ETC.
15,14
11,10
8,7
0,15,14
11,10,9
2,1,0, 15
0
INSTRUCT
CODE
EXTERNAL SYMBOL I.D. NO.
INSTRUCT
CODE
EXTERNAL SYMBOL I.D. NO. –OR– ZERO
M R
UNRELOCATED VALUE –OR– OFFSET
D/1
D/1
INSTRUCTION WORD
R = 100
INSTRUCTION WORD
R = 101
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
15
12 11
2 10 15
0
TYPE
M R
RELOCATABLE BYTE ADDRESS
INSTRUCTION WORD R = 110
TODS—17

END RECORD

CONTENT





EXPLANATION

RECORD LENGTH = 4 WORDS
IDENT = 101

R: RELOCATION INDICATOR FOR TRANSFER ADDRESS

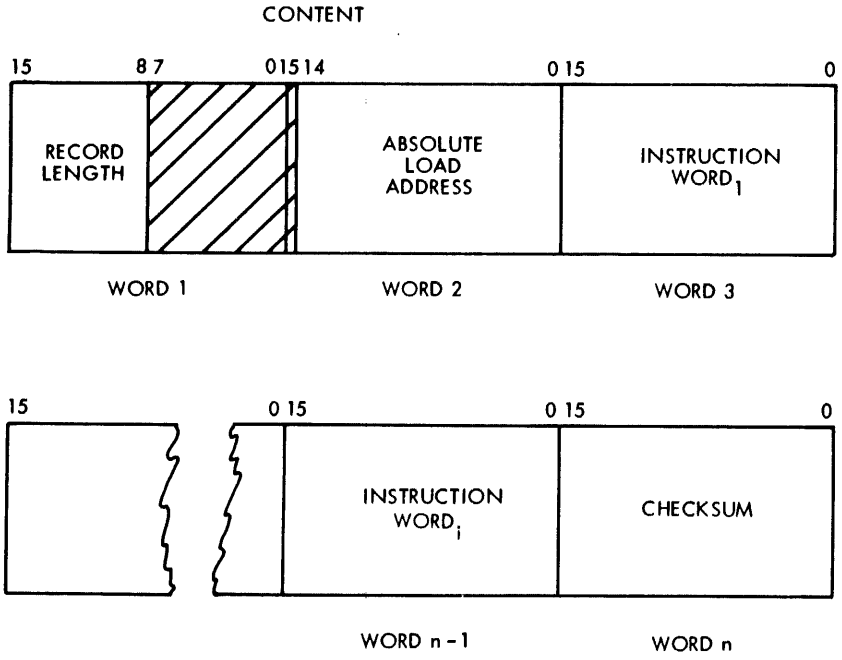
= 0 IF PROGRAM RELOCATABLE
= 1 IF BASE PAGE RELOCATABLE
= 2 IF COMMON RELOCATABLE
= 3 IF ABSOLUTE

T: TRANSFER ADDRESS INDICATOR

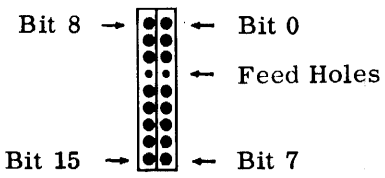
= 0 IF NO TRANSFER ADDRESS IN RECORD
= 1 IF TRANSFER ADDRESS PRESENT

TODS—18

ABSOLUTE TAPE FORMAT



† On paper tape, each word represents two frames arranged as follows:



APPENDIX I
RTE-II VS RTE-III

RTE-III attempts to provide a user program environment the same as that in RTE-II. Most application programs developed for RTE-II and interfacing with the system only through documented, standard functions will run correctly under RTE-III. This appendix summarizes exceptions to this rule and other points of interest to RTE-II users.

PRIVILEGED DRIVERS

RTE privileged drivers must save and restore the state of the machine at interrupt. In RTE-II this includes the A,B, E and 0 registers, the point of interrupt, and the state of memory protect. In RTE-III the state of the machine includes these items plus the x and y registers and the status of the Dynamic Mapping System. In addition, if the privileged driver must access a user buffer which is in the program area, it must save, load, and restore the user map. See Section V.

$REIO WITH DOUBLE BUFFERING

In RTE it is possible for a program to make a double-buffered $REIO call. $REIO moves only the first buffer. In RTE-II the other buffer can remain in the program area or it can be separately moved to system available in memory with a $LIBR call. In RTE-III it is the user's responsibility to see that the second buffer is moved to System Available Memory.

COMMUNICATIONS AREA

Some of the words in the base page communications area that are static in RTE-II, are dynamic in RTE-III. Some of these are BKDRA, BKLWA, RTDRA, AVMEM, and FENCE. In RTE-III, these words are reset for each new program which is dispatched. See Appendix A for a complete list.

COMMON USAGE

A disc or memory resident program using the Real-Time or Background Common areas must declare common in order to have it included in its User Map. In addition, the total common area to be accessed must be declared in the main module for programs loaded at generation, or in the first relocatable module input to the loader for programs loaded on-line. In RTE-II it was possible for main programs and their segments to access different common areas (e.g. one using BG common, one using RT). This is not possible in RTE-III; type and size of common is determined once for the entire program. In RTE-II a Background program using Real-time (foreground) common could read but not modify that area; in RTE-III the fence is set to allow full access to the selected common area.

FIXED-HEAD DISCS

Fixed-head discs are supported in the RTE-II System. They are not supported in the RTE-III System.

DEFAULT COMMON FOR ON-LINE RELOCATION

If system common or reverse common is not specifically requested, local common is used without regard to program type. Refer to Section IV, Part 5, RTE-III Relocating Loader.

INTERRUPT TRAP CELLS

Due to DMS characteristics, instructions other than JMP and JSB must not be placed in a device's interrupt location. Execution of a non-transfer instruction (NOP,CLC,HLT,etc) may cause the interrupted program to continue under the wrong DMS Map. The privileged card (12620A only) is not a device and does not interrupt.

PROGRAM TYPES

Table I-1 is a chart summarizing the RTE-III program types. Note that several types have been added while 3 have been obsoleted.

The program type is used during generation to indicate various program characteristics. Program type is not used for programs loaded on-line; desired characteristics are specified through parameters to the loader.

RTE-III allows the user to expand the program's primary type code to cover such features as access to real-time common by background programs and access to the subsystem global area (SSGA). The primary type code is expanded by adding 9,16, or 24 to the original number. For example, if you have a background disc resident program with a primary type code of 3 (it accessed background common or no common at all), and you wanted to modify it to access SSGA, you would change the type code to 19 (add 16 to 3).

Table I-1. Summary of RTE-III Program Types

PROGRAM CATEGORY / EXECUTABLE PROGRAMS	PROGRAM TYPE	COMMON ACCESS: REAL TIME COMMON	BACKGROUND COMMON	SSGA	RT COMMON & SSGA	BG COMMON & SSGA	LOAD POINT: NO COMMON DECLARED	SOME COMMON DECLARED	MEMORY PROTECT FENCE: NO COMMON DECLARED	SOME COMMON DECLARED
MEMORY RESIDENT*	1	✓					L_1	L_1	F_5	F_3
	9		✓				L_1	L_1	F_5	F_4
	17			✓			L_1	L_1	F_1	F_1
	17				✓		L_1	L_1	F_1	F_1
	25					✓	L_1	L_1	F_1	F_1
REAL TIME DISK RESIDENT*	2	✓					L_3	L_2	F_2	F_3
	10		✓				L_3	L_2	F_2	F_4
	18			✓			L_2	L_2	F_1	F_1
	18				✓		L_2	L_2	F_1	F_1
	26					✓	L_2	L_2	F_1	F_1
BACKGROUND DISK RESIDENT*	3		✓				L_3	L_2	F_2	F_4
	11	✓					L_3	L_2	F_2	F_3
	19			✓			L_2	L_2	F_1	F_1
	19					✓	L_2	L_2	F_1	F_1
	27				✓		L_2	L_2	F_1	F_1

SPECIAL PROGRAMS	TYPE	DESCRIPTION
SYSTEM MODULE	0	MODULE TO BE LOADED WITH RESIDENT SYSTEM. PART OF HP SUPPLIED SYSTEM, USER-WRITTEN DRIVER, ETC.
BACKGROUND SEGMENT	5	OVERLAYABLE PROGRAM USED WITH BG DISK RESIDENT MAIN. COMMON TYPE, FENCE ADDR, AND LOAD PT. DETERMINED BY MAIN.
SUBROUTINE	6	RELOCATED INTO RESIDENT LIBRARY IF CALLED BY ANY MEMORY RESIDENT PROGRAM. (UNCALLED 6'S BECOME 7'S)
SUBROUTINE	7	STORED ON DISK IN RELOCATABLE FORM. ANY PROGRAM CALLING A TYPE 7 HAS A COPY APPENDED TO IT.
SUBROUTINE	8	APPENDED TO CALLING PROGRAM. ALL TYPE 8 RELOCATABLES ARE DISCARDED AFTER GENERATION.
SUBROUTINE	14	RELOCATED INTO RESIDENT LIBRARY, WHETHER CALLED OR NOT. (FORCE LOADED)
SSGA MODULE	30	RELOCATED INTO SUBSYSTEM GLOBAL AREA OF SYSTEM. ACCESSIBLE ONLY TO PROGRAMS OF PROPER TYPE (ABOVE)
(OBSOLETE)	4	CONVERTED TO TYPE 9 WHEN ENCOUNTERED. (DEFINED AS BG CORE RESIDENT W/BG COMMON IN RTE-II)
(OBSOLETE)	12	CONVERTED TO TYPE 1 WHEN ENCOUNTERED. (DEFINED AS BG CORE RESIDENT W/RT COMMON IN RTE-II
(OBSOLETE)	13	CONVERTED TO TYPE 5 WHEN ENCOUNTERED. (SEE TYPE 5) (DEFINED AS BG SEGMENT USING RT COMMON IN RTE-II)

LOAD POINT & FENCE DEFINITIONS (SEE FIGURE 6-3)

L_1 – NEXT AVAILABLE LOCATION DURING LOADING OF RESIDENTS.
L_2 – 3RD WORD OF NEXT PAGE AFTER COMMON AREAS.
L_3 – 3RD WORD OF NEXT PAGE AFTER RESIDENT LIBRARY.

F_1 – FIRST WORD OF SSGA.
F_2 – FIRST WORD OF PAGE FOLLOWING RESIDENT LIBRARY.
F_3 – FIRST WORD OF RT COMMON
F_4 – FIRST WORD OF BG COMMON
F_5 – FIRST WORD OF RESIDENT PROGRAM AREA.

* ADD 80 TO ANY OF THESE TYPES TO SPECIFY AUTOMATIC SCHEDULING AT SYSTEM STARTUP.

INDEX

INDEX (Continued)

INDEX (Continued)

INDEX (Continued)

L (Continued)

M

N

O

O (Continued)

P

INDEX (Continued)

INDEX (Continued)

READER COMMENT SHEET

Real-Time Executive III Software System
Programming and Operating Manual

92060-90004 JUL 1976

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?

Is this manual complete?

Is this manual easy to read and use?

Other comments?

FROM:

Name _____

Company _____

Address _____

_____

_____

FOLD

FOLD

FIRST CLASS
PERMIT NO. 141
CUPERTINO
CALIFORNIA

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States Postage will be paid by

Manager, Technical Publications
Hewlett-Packard Company
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

FOLD

FOLD

PART NO. 92060-90004
Printed in U.S.A. 7/76

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
11000 Wolfe Road, Cupertino, California 95014