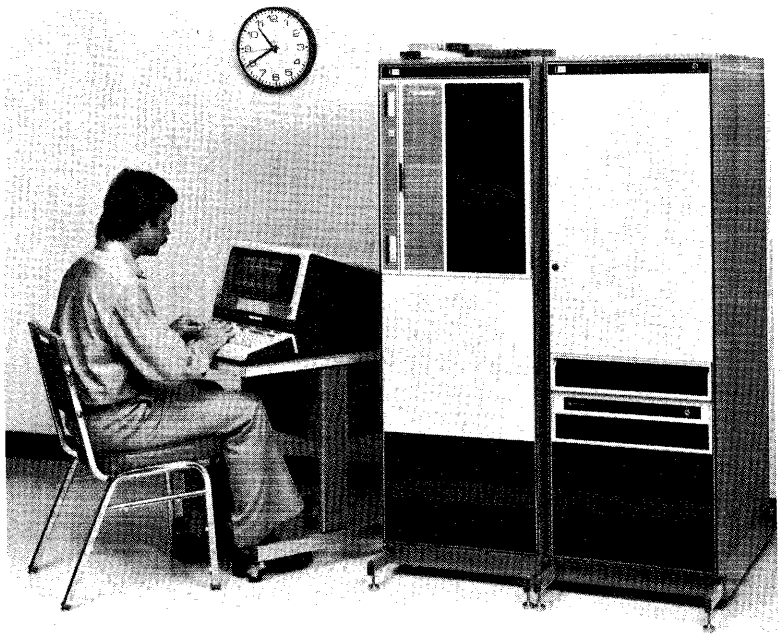
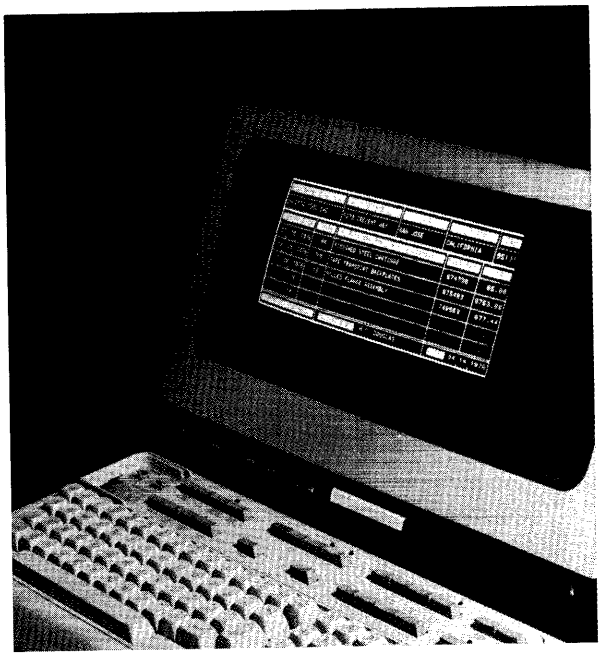
HEWLETT hp PACKARD

RTE-III

General Information Manual





RTE-III

General Information Manual

HEWLETT PACKARD

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

Library Index Number
2RTE.310.92060-90009

PART NO. 92060-90009
PRODUCT NO. 92060

Printed in U.S.A. 2/76

NOTICE

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

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

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

PREFACE

This manual presents the basic capabilities of the HP RTE-III Real-Time Executive Operating System to prospective and new users of this HP product.

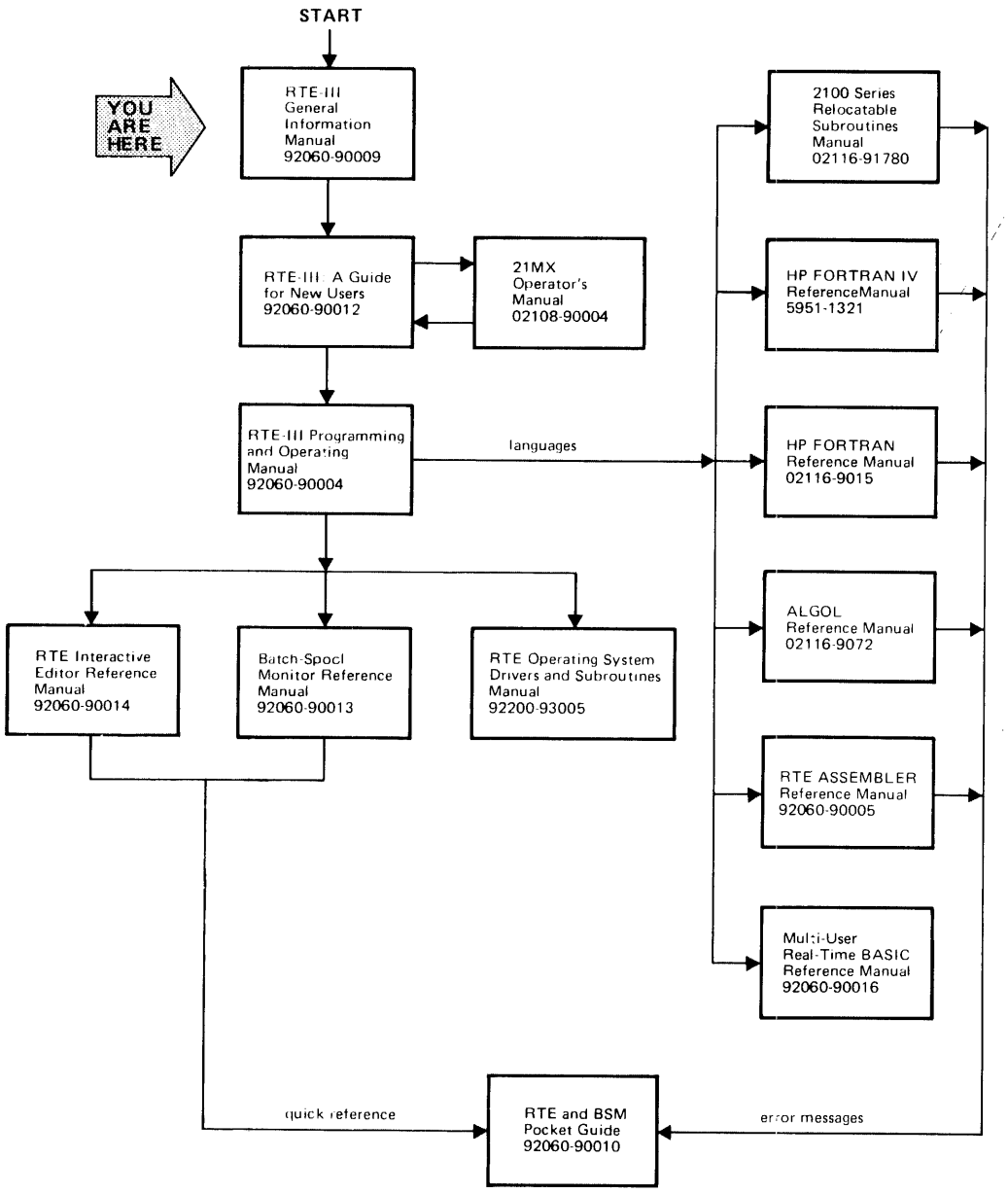
- Section I presents some basic concepts of operating systems and real-time operating systems.
- Section II presents the HP RTE-III Operating System, a Hewlett-Packard approach to the concepts presented in Section I. The basic capabilities of RTE-III are discussed. In addition, RTE-III terminology is defined.
- Section III discusses how RTE-III manages main memory.
- Section IV lists and briefly describes RTE-III system commands including operator commands and program calls to the system executive.
- Section V contains a list and brief description of the Batch-Spool Monitor operator commands and program calls.

This manual presents the concepts and capabilities of RTE-III in general terms. The following manuals provide detailed information about RTE-III:

- HP RTE-III: A Guide for New Users (Part No. 92060-90012). This guide presents the fundamentals for using RTE-III. You may use it as a workbook which provides examples that you can try from a terminal as you read the explanation of system features.
- HP RTE-III Programming and Operating Manual (Part No. 92060-90004). This manual provides RTE reference material and experienced programmers may use it as a programming guide to the RTE-III system.
- HP RTE-III Interactive Editor Reference Manual (Part No. 92060-90014). This is the reference manual and programming guide to the RTE Interactive Editor (EDITR).
- Batch-Spool Monitor Reference Manual (Part No. 92060-90013). This is the reference manual and programming guide to file management, batch processing, and I/O spooling in the RTE-III Operating System.

The following documentation map gives the titles and part numbers of the complete set of RTE-III manuals and illustrates their relation to each other.

DOCUMENTATION MAP



CONTENTS

ILLUSTRATIONS

TABLES

WHAT IS AN OPERATING SYSTEM?

SECTION I

The principal function of an operating system is to manage the use of system resources by your programs. Your programs compete for use of system resources such as central processor (CPU) time, main memory, disc storage, input/output devices, and system library software. You assign a priority value to each of your programs. The operating system uses this value to determine which program among those competing for a specific resource will actually be given access to that resource. The program with the highest priority value is granted access to the resource first.

An operating system may be defined as an organized collection of programs which increases the productivity of a computer by providing control and management capabilities together with common functions shared by user programs. Once the computer system is generated and running, the operating system takes responsibility for control of the system environment.

Included among the major functions provided by an operating system are:

- SYSTEM EXECUTIVE (Resource Allocation)

  The system executive controls the operating system environment by allocating resources such as main memory, central processor (CPU) time, input/output, and mass storage. Communication between your programs and the system is handled through calls to the executive.

- SCHEDULING (Program Scheduling)

  The system scheduler organizes and schedules your programs for execution. The scheduler activates, terminates, suspends, or prepares programs for execution.

- I/O CONTROL (Input/Output Scheduling and Transfer)

  Two important tasks of I/O control are scheduling of I/O and the transfer of data.

  For scheduling, the operating system accepts the request for I/O and sets up the operation necessary to perform the requested action.

  The transfer of data (that is, the movement of data between main memory and I/O devices or mass storage) is handled by I/O "driver" programs which support the I/O devices associated with your system. The I/O device driver sets up the operation conditions, flags and buffers and then activates the I/O device.

- DATA MANAGEMENT

  The data management functions include file management and I/O support.

  File management provides for the creation of new data files and the modification or elimination of existing files. In addition, security is provided for your files to prevent unauthorized access.

The I/O support function provides for uniform and consistent access to the I/O devices for your programs.

- SYSTEM GENERATION

An operating system may include generation routines that allow you to create an operating system tailored to your hardware and application requirements. That is, a master operating system can be customized during system generation to support your configuration. You select the necessary modules including support programs such as language compilers, utility programs, and data management routines that are required specifically for your application.

WHAT IS A REAL-TIME OPERATING SYSTEM?

A real-time operating system may be defined as one which controls its environment by reviewing data, processing that data, and taking action with sufficient speed to affect immediately the functioning of the environment.

- MULTIPROGRAMMING

A real-time operating system should be capable of concurrent or parallel execution of more than one program.

- MEMORY MANAGEMENT

Real-time main memory must be managed so that the most efficient use of memory is maintained. Memory is used to execute programs, to accept data from input devices, and to hold data to be transferred to output devices. The operating system must monitor the use and allocation of memory just as it would any of the system resources.

- MULTIPLE MEMORY PARTITIONS

Partitioning of memory is used to allow more than one program to be resident in main memory at a time. This decreases the amount of time required to switch execution from one program to another. The system can switch execution between programs within main memory from 50 to 100 times faster than when a program must be transferred into main memory from disc storage.

- PROGRAM SCHEDULING BY PRIORITY

In real-time computer systems, some events must be acted upon without delay. This can be a problem if two or more events simultaneously demand attention. One solution is to assign an execution priority to every program in the system. Then, if several programs are competing for the central processing unit, execution will occur in order of program priority.

THE HP RTE-III OPERATING SYSTEM

SECTION II

The HP RTE-III Real-Time Executive Operating System controls a computer system based on the HP 21MX Computer series with expanded memory.

RTE-III is a multiprogramming operating system. Programs are scheduled for execution by priority. Through a unique memory mapping system,utilization of up to 256K words of main memory is possible. Parallel execution of two or more programs is accomplished using these features. The minimum amount of main memory required for operation is 32K words.

RTE-III is a real-time multiprogramming operating system which provides the following features:

- System Generation and Installation
- Operator and Program Control of RTE-III
- Multiple Language Support
- Program Management
- Multiprogramming
- Memory Partitioning
- Resource Management
- Input/Output Management
- Multi-Terminal Operation
- Program Development
- Distributed Systems Support

SYSTEM GENERATION AND INSTALLATION

You generate your system using the same hardware on which the system will operate. You use a generation program (RTGEN) to perform the initial system generation. Using RTGEN, you incorporate RTE-III Executive modules together with library routines and your programs into a real-time operating system. The system modules reside in main memory. Your programs may be either main memory resident or disc resident. You can declare routines that require immediate response to real-time conditions or small programs that must execute often as memory resident programs, saving critical load time. Because disc resident programs must be loaded into a memory partition at least once, they take longer to run to completion.

Before executing RTGEN, you should plan the structure of your system. Disc and I/O planning worksheets are included in the RTE-III Reference Manual for this purpose. Once you formulate the system plan, you use generator input worksheets to record the information required by RTGEN. The generator worksheets are in the form of the actual questions which RTGEN will display on the system console. You need only write the appropriate response on the blank line provided for each question.

Once RTGEN begins execution, you can follow the displayed dialog with your input worksheet and enter the information from the console keyboard. When you have completed entering the appropriate responses, RTGEN stores the configured system on the system disc and reports this fact to you together with the size of the system on the disc tracks. At this point you can start up the system using the procedure outlined in the RTE-III Reference Manual.

CONTROL OF RTE-III

You control the RTE-III system through operator commands entered from a system console device (or auxiliary terminal) or through system commands included within your programs.

OPERATOR CONTROL

Operator control over the RTE-III operating system is exercised using a set of RTE-III operator commands. These commands are used interactively (see Section IV). The system issues an appropriate response to each command you enter. The response may be a message or simply a line feed. Additional operator commands are provided by the Batch-Spool Monitor, a subsystem of RTE-III. Batch-Spool Monitor supplies extensive file management and I/O spooling capabilities (see Section V). You may use all of these operator commands interactively to monitor and control RTE-III activity from several terminals via the Multiple Terminal Monitor, another subsystem of RTE-III (see Multi-Terminal Operation).

PROGRAM CONTROL

In addition to operator control, your programs can include calls to the system executive module, EXEC (see Section IV). The EXEC calls are the communication link between each executing program and RTE-III. The Batch-Spool Monitor increases program control capabilites by supporting calls to the File Management Package and the Spool Monitor (see Section V).

LANGUAGE SUPPORT

RTE-III provides several programming languages and a relocatable subroutine library. The languages provided are:

- Real-Time HP 21MX Assembler
- Real-Time HP FORTRAN IV Compiler
- Real-Time HP FORTRAN Compiler (compiles FORTRAN II programs)
- Real-Time ALGOL Compiler
- Optionally, the HP Multi-User Real-Time BASIC software can be added to RTE-III

The relocatable subroutine library includes mathematical subroutines and utility subroutines. You can use these routines in a shared (re-entrant) mode or append them to your program.

PROGRAM CATAGORIES

There are three program categories within RTE-III:

- Memory Resident Programs
- Real-Time Disc Resident Programs
- Background Disc Resident Programs

Each of these categories contains several program types depending on the type of common used. There is a special program category for system modules, subroutines, background segments, and Subsystem Global Area (SSGA) modules.

PROGRAM IDENTIFICATION

Before any program can be executed, it must be known to the system. RTE-III maintains an ID segment for each of your programs loaded during system generation. In addition, during system generation, you should declare a specific number of blank ID segments; one for each program you intend to load during operation of the system.

When a program is loaded on line, RTE-III assigns a blank ID segment to that program. This segment identifies and controls program linkages until the program is removed from the system. The ID segment then becomes available to another program.

The ID segment is the program identifier. It contains static information about the program such as the program name, priority (priority can be changed by an operator command), primary entry point, and so forth. RTE-III maintains dynamic information in the ID segment such as a list link word, point of program suspension, track and sector location on disc, and so forth. A main program requires a 28-word ID segment. A program segment requires a 9-word ID segment.

PROGRAM AREAS

There are two areas within main memory where your programs execute, the memory resident program area and the disc resident program area.

The memory resident program area must exist within the first 32K words of physical main memory. This area contains all of your memory resident programs which are loaded during system generation. It immediately follows the common area within main memory.

The partitioned portion of main memory is the disc resident program area, that is, the area of main memory in which your disc resident programs will execute. You can load disc resident programs during system generation and on line during normal system operation. For on line program loading, use the RTE-III Relocating Loader (LOADR). You may declare disc resident programs either permanent or temporary programs.

PERMANENT PROGRAMS

Because a permanent program ID segment exists in the system area of the disc and is loaded into main memory with the system, permanent disc resident programs are always available to RTE-III even through a system shut down and subsequent start up. The permanent program ID segment is saved on disc through system shut down and restart.

You create permanent programs during system generation and with LOADR. You can purge (delete) permanent programs from the system only by using LOADR or by regenerating the system.

TEMPORARY PROGRAMS

Temporary disc resident programs are lost to the system following a shut down. A temporary program ID segment exists only in main memory and its information is lost whenever the system is shut down except for a power fail/automatic restart operation. You can purge (delete) temporary programs from the system using an RTE-III operator command, OFF (see Section IV).

PROGRAM STATES

In a running RTE-III system, there are four general program states:

- Executing
- Scheduled for Execution
- Suspended from Execution
- Dormant

At a given instant, only one program is executing. RTE-III maintains a list of programs scheduled for execution and a list of programs suspended from execution. The WHZAT program described in *RTE-III: A Guide for New Users* (Part No. 92060-90012) can be executed to display the scheduled and suspended programs and their status. Programs that have run to completion or that are not executing and not in the scheduled or suspended lists are in the dormant state. Figure 2-1 illustrates these lists and programs states.

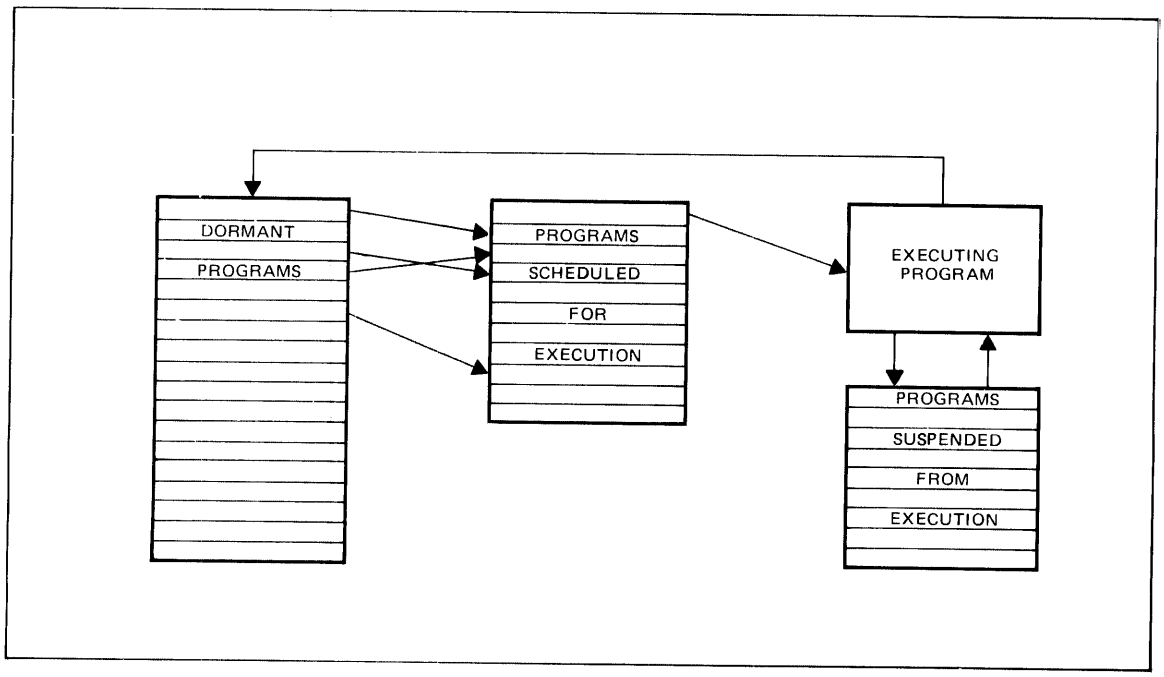


Figure 2-1. RTE-III Program States

SCHEDULING PROGRAMS

Programs can be scheduled for execution by:

- External Event Interrupt
- Operator Request
- Program Call to EXEC
- Time Intervals on the System Real-Time Clock

PROGRAM EXECUTION

For programs in the scheduled list, execution begins immediately for the program having the highest priority. For example, if program ONE is executing when program TWO with a higher priority is scheduled, program ONE is suspended from execution while program TWO executes.

REAL-TIME PROGRAMS

These programs execute in response to an event that must be acted upon immediately. The action taken by the program must be quick enough to affect the current status of the event.

The most critical real-time programs should reside in main memory where they can be executed without delay. Programs of less critical nature can reside on disc and will be loaded or swapped, if necessary, into a main memory partition from the disc.

BACKGROUND PROGRAMS

The background disc resident programs generally have the lowest priority in the system. These programs are run during that part of CPU time not used by higher priority programs. Program development (editing, compiling, debugging, and so forth) is an example of background program operation.

SCHEDULING PROGRAM EXECUTION

Under real-time computer control, some events must be acted upon without delay. This can be a problem if two or more events simultaneously demand attention. The solution is to assign an execution priority to every program in the system. Then, if several programs are competing for the central processor, they are executed in order of priority.

MULTIPROGRAMMING

Multiprogramming is the parallel execution of two or more programs which compete for system resources within a single central processor unit. For example, while the execution of one program is suspended for completion of an I/O operation, another program executes. Program execution occurs at such speed as to appear concurrent.

In a batch processing computer system, you submit the programs to be executed to the system one after the other. The first program entered into the system is the first program executed. The second program entered into the system cannot execute until the first program completes execution.

In a time-sharing computer system, the execution of two or more programs is interleaved so that each program gets an equal portion of CPU time in which to execute.

In a multiprogramming computer system, the execution of two or more programs is performed in parallel (or concurrently). Because a single CPU is capable of executing only one program at a time, a scheme must be devised which permits many programs to be prepared for execution and then to interleave the execution of those programs. For example, suppose that three programs (named PROG1, PROG2, and PROG3) reside in main memory and are ready to execute. All system resources are available to each program. A priority scheme is used to interleave program execution — PROG1 has the highest priority and PROG3 has the lowest priority. The operating system passes control to PROG1 first. PROG1 has control of the system until it completes execution or until it must wait for a system resource such as an I/O operation. When PROG1 goes into a wait cycle, control is passed to PROG2 (having the next lower priority) which executes until it terminates, until it must wait for a system resource, or until PROG1 is again ready to execute. If PROG2 terminates or goes into a wait cycle, then control is given to PROG3. If PROG1 again becomes ready to execute control is given to PROG1 because control always goes to the program having the highest priority which is ready to execute.

Now, suppose that PROG1, PROG2, and PROG3 each have the same priority. PROG1 is in execution and goes into a wait cycle. Control is given to PROG2 and PROG1 is placed in the wait queue behind PROG3.

Figure 2-2 illustrates the execution of separate programs in batch processing, time-sharing, and multiprogramming environments.

MEMORY PARTITIONING

The portion of main memory remaining after the RTE-III operating system is loaded (including main memory resident programs), is divided into blocks of pages (1,024 words per page) called partitions. The disc resident user programs (your programs) execute in these partitions. This scheme reduces program transfers (swapping) between disc and main memory since as many of your disc resident programs as there are available partitions can be present in main memory ready for execution. You can declare a maximum of 64 numbered, fixed-length partitions within a physical memory of up to 256K words.

You define the partition size and partition number during system generation. At the same time, you may declare a partition class of real-time or background. This allows you some control over which programs compete for partitions in which to execute because, by default, real-time programs execute in real-time partions and background programs in background partitions. However, you may assign a program to a specific partition regardless of partition type. In addition, you can reserve a partition for execution of programs assigned to it, excluding other programs from using that partition.

Partitions always begin and end on a page boundary. Each partition must have at least two pages: a base page (communications area for system and program linkages) and one additional page for disc resident program execution. Above this minimum, partition size depends on

program requirements. During system generation, RTGEN reports the size of each program in the system. Programs which use dynamic buffering (for example, the Interactive Editor) require additional pages within a partition for the buffers. You must allow for these extra pages at generation time. You use the information reported by RTGEN to determine the size of the partitions. Normally, partitions will be between 2 and 18 pages in length.

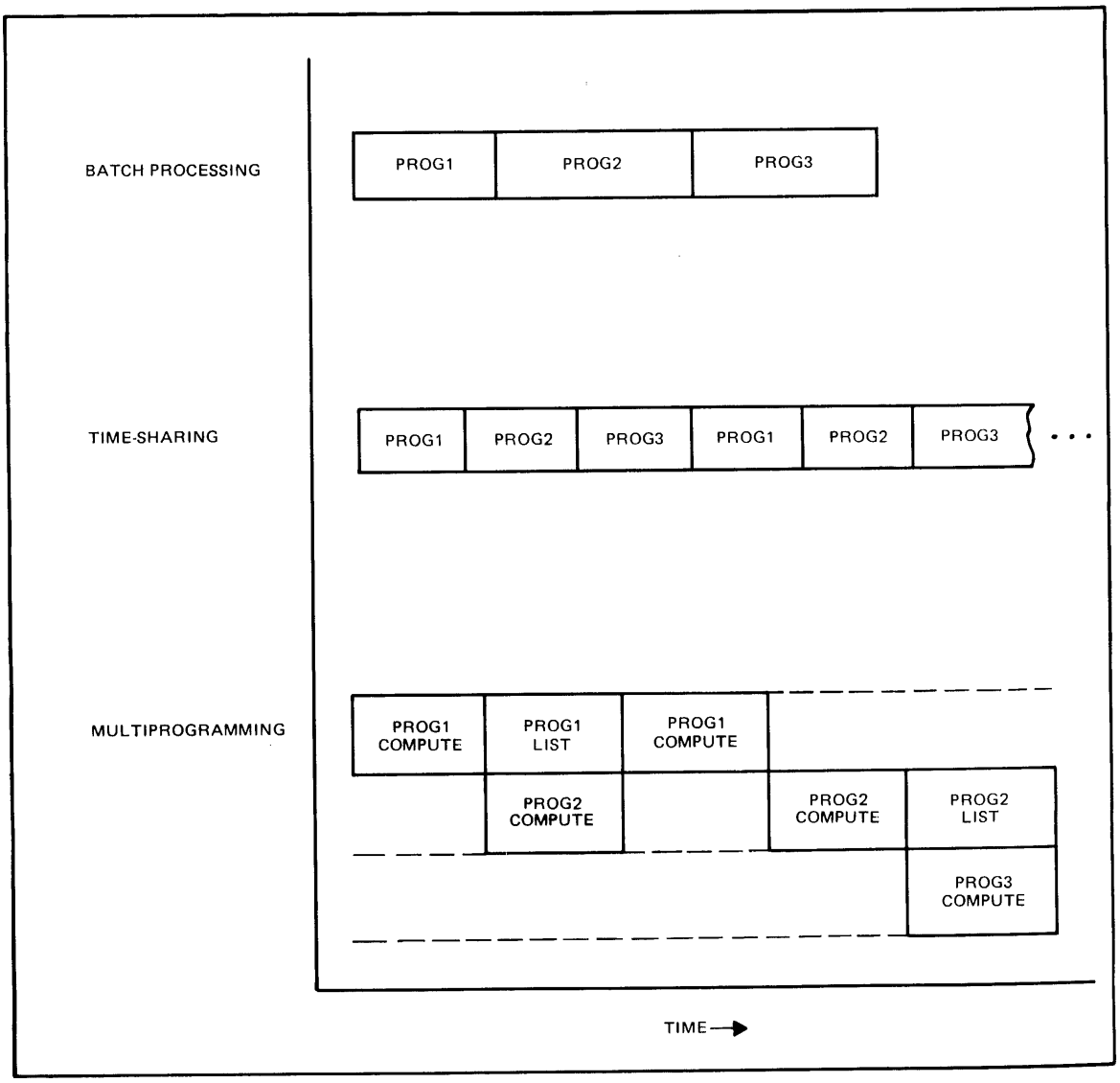


Figure 2-2. Program Execution Flow

RESOURCE MANAGEMENT

Your programs compete for use of system resources such as central processor unit (CPU) time, main memory, disc storage, and input/output devices. You assign a priority value to each of your programs. The system uses this value to determine which program among those competing for a specific resource will actually be given access to that resource. The program with the highest priority value executes first.

Another level of resource management exists which provides "user defined" resource sharing. That is, you determine which of your programs will share a resource. You accomplish this through a resource numbering call to the system executive module. For example, you may wish to share a resource such as an I/O device, file, or subroutine between several programs and require that only one of your programs have access to the resource at any time. One program issues the call to allocate a resource number. Following allocation of the resource number, any of the cooperating programs can issue a call to lock the number for exclusive use of the resource associated with that number.

A resource number is not a physical entity, nor is it assigned logically to a resource. The RTE-III system is always aware of the resource number but not aware of the resource with which it is associated. The programs sharing the resource number must agree on which resource is associated with which number.

Resource number (RN) locking allows two or more programs to access sensitive areas of their own code on a "one at a time" basis. Figure 2-3 illustrates this process. Assume PROGA obtains the locked RN first. In this case, PROGB is suspended until PROGA unlocks the RN at which time PROGB is reactivated. Then, PROGB locks the RN which prevents PROGA (or any other program) from obtaining a lock on the same RN. PROGA and PROGB cooperate in their use of the RN. That is, both programs agree to use the same RN and may control more than one RN to protect other areas of code.

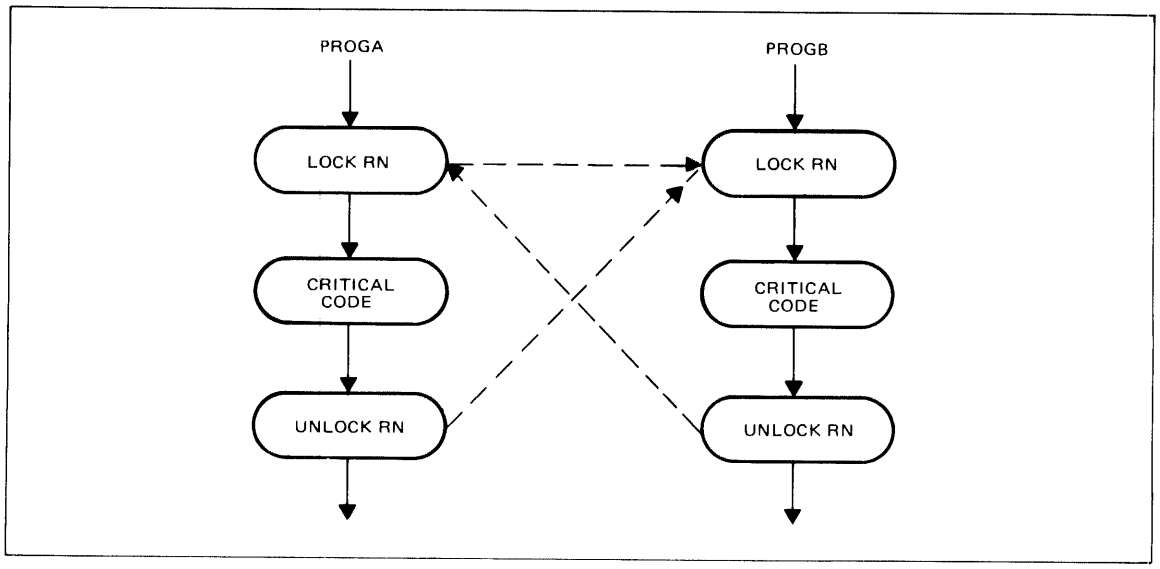


Figure 2-3. Resource Number Sharing

INPUT/OUTPUT

An RTE-III system module handles the scheduling and control of your program I/O requests. This centralized scheduling and control feature together with logical referencing of physical I/O devices provides programming capabilities independent of the system I/O devices. That is, your executing program accepts input data, processes it, and releases output data without concern for which type of I/O device is handling the data.

Included among the automatic functions of the I/O scheduling and control module is the maintenance of an I/O wait list. If you issue a request for I/O to a busy device (the device is handling some other I/O operation), your request is placed in a waiting list in order of your programs priority. Your I/O request will be processed when the device becomes not busy and when your request has the highest priority in the wait list for this device. Execution of your program is suspended while the I/O request is in the wait list.

Optional I/O functions available include:

- Device Time-Out

  When you generate your RTE-III system, you can set a time-out value for any I/O device. Setting a time-out value for a device prevents your program from being suspended indefinately because of an incomplete I/O operation. For example, a hardware malfunction could prevent the return of an "I/O completed" signal. In this case, when the specified time interval has elapsed, the system will place the I/O device in "down" status and transmit a warning message to the system console. A down device cannot be accessed for I/O operations until you correct the situation. In addition to system handling of device time-out, you can code the I/O device driver software to direct the action to be taken when a time-out condition is encountered.

- I/O Buffering

  You may select automatic I/O buffering on low or medium speed devices. When you select this capability your programs can initiate an output operation and proceed without waiting for completion of the I/O operation. For input buffering, see "Class I/O".

- Re-entrant I/O

  You can choose to have your I/O processing operations re-entrant. In this case, your disc resident programs may be swapped to disc storage even if you are suspended for I/O. This allows programs with a higher priority to use the partition and I/O device which were associated with a program having a lower priority.

- Exclusive Logical Unit Assignment

  Temporarily, you can assign a logical unit to your program. This prevents any other program from accessing the assigned logical unit until you unlock it.

- Class I/O

  Program-to-program communication normally occurs through the system common area. The class I/O function provides you with a method of program-to-program communication other than through the system common area.

You can accomplish program-to-program communication by using a class WRITE/READ call. There is no I/O device associated with this call, logical unit zero is assumed. Data is written into the class buffer by one program and read from the class buffer by one or more other programs. The class WRITE/READ calls use a "mailbox" scheme for data transfers. One program places data in the mailbox (class buffer) and cooperating programs can access the mailbox to retrieve the data. Figure 2-4 is a sample of communication between two programs.

In addition to program-to-program communication, data transfers between programs and multiple I/O devices is possible using class I/O.

A class is a numbered account associated with a buffer area in system available memory. You establish class numbers (up to 256) during system generation. When a program issues an initial class I/O call, the system assigns a class number to the calling program and allocates a buffer. The calling program can perform class I/O operations and pass the class number to other programs which, in turn, can use the class number to make class I/O calls and pass the class number to other programs. Any of the cooperating programs can release (deallocate) the class number back to the system.

For example, program A requests a class READ operation to read data from an input device into the class buffer. Then program A issues a call to EXEC to schedule program B for execution, passing the class number to program B in the parameter list. Program B requests a class GET operation to obtain the program A data from the class buffer. The GET call can be coded to either retrieve the data and destroy the contents of the buffer, or retrieve the data and leave the buffer contents intact. In the latter case, other programs can access the same data. Following execution of any class I/O call, the A-register contains the completion status of the call.

You may use a similar procedure to call a class WRITE operation. A class WRITE operation transfers program data through the class buffer to an output device designated by a logical unit number. The program continues to execute. You must issue the class GET call to check the completion status.

MULTI-TERMINAL OPERATION

RTE-III supports the simultaneous operation of more than one terminal device in addition to the system console through the Multiple Terminal Monitor (MTM) package. Users at these terminals have access to the system and to disc files containing programs or data. Several users at separate terminals can manipulate files and perform edit operations at the same time. Compile, assemble, or load operations also can be performed from any of the multiple terminals, however, only one user at a time should attempt these operations because there is only one logical source (LS) and one load-and-go (LG) track area from which to work.

The entire set of supported languages is available to you in the multiple terminal environment. Whenever you need to include a new program or to change programs within your RTE-III system, you can accomplish the program development work and loading on line. The normal real-time operation of your system need not be interrupted. The RTE-III Assembler, FORTRAN compilers, ALGOL compiler, and Multi-Terminal Real-Time BASIC are available as programming languages. In addition, the Interactive Editor (EDITR) is available. EDITR is used to edit program and data files which reside on disc storage.

```
FTN,L
      PROGRAM PROGA
      DIMENSION IBFR(32),INAME(3)
        :
C
C     DO CLASS WRITE/READ TO LU ZERO.
C
      ICLAS=Ø
      CALL EXEC(2Ø,Ø,IBFR,-64,IDUMY,JDUMY,ICLAS)
C
C     SCHEDULE PROGB AND PASS CLASS NUMBER.
C
      INAME(1)=5Ø122B
      INAME(2)=475Ø7B
      INAME(3)=41ØØØB
      CALL EXEC(1Ø,INAME,ICLAS)
        :
      END
```

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

Figure 2-4. Program-to-Program Communication

On completion of program development, you can use the on line loader (LOADR) to prepare the program for loading into a partition for execution or debugging. Figure 2-5 illustrates multiple terminal operation.

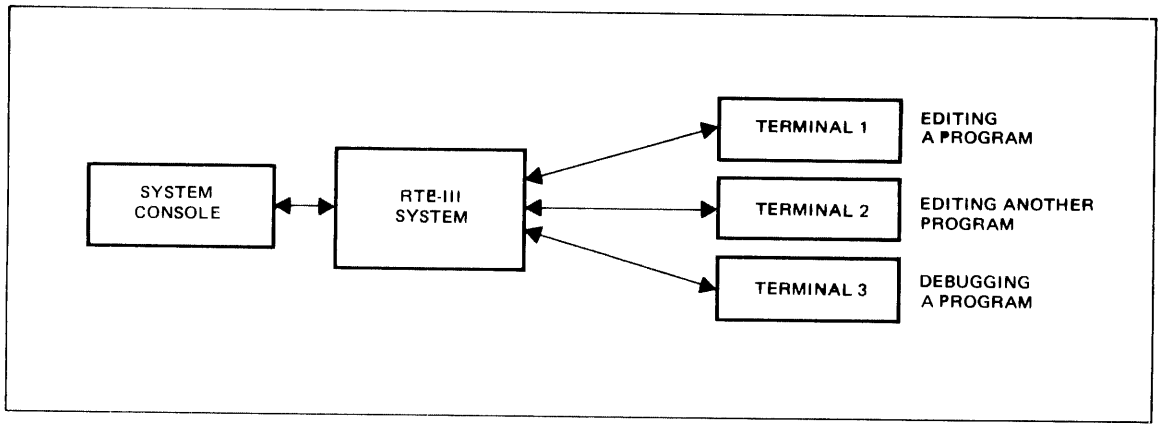


Figure 2-5. Multiple Terminal Environment

DISTRIBUTED SYSTEMS

The term "distributed systems" encompasses a set of compatible hardware and software packages (interfaces plus communications software) that support network communications between a central computer system and one or more satellite computer systems.

CENTRAL SYSTEM

The central system is based on the HP RTE-II or RTE-III operating system. Central mass storage and peripheral equipment capabilities together with network monitoring functions are provided.

SATELLITE SYSTEMS

The distributed satellites are based on a variety of HP Operating Systems such as RTE-II, RTE-III, RTE-B, and RTE-C. The satellite systems have access to mass storage at the central system. Using remote file access (RFA) calls to the RTE File Management Package, you (at a satellite system) can create, rename, open and close files, read and write to specific records, or obtain the status of a file at the central site.

Remote EXEC calls include functions to schedule program execution at the central site, load programs from central into a satellite for execution, send messages between systems, and access the central real-time clock.

Program-to-program communication provides for data transfer between application programs in the satellite and central systems. This is accomplished via a "master/slave" program relationship. The master program in either the central or a satellite system assumes responsibility for initiating the data link and data transfers between the programs. The slave program operates much like a peripheral device.

You can develop new programs and then store them at the central site. The central system can then force a "down load" of the program to any satellite in which the program can execute. This forced down load capability allows satellite systems to run unattended by an operator.

Further, you can establish a communication link with computer systems outside the distributed system environment. You can exchange data with such computer systems as the HP 3000, another HP RTE-III system, or an IBM 360/370 system. Figure 2-6 shows the relationship of the distributed system computers.

Information about HP Distributed System software is contained in the following manuals:

HP Distributed System CCE Central Communication Executive, Programming and Operating Manual, part no. 91700-93001.

HP Distributed System SCE-1 Satellite Communication Executive, Programming and Operating Manual, part no. 91700-93003.

HP Distributed System SCE-3 Satellite Communication Executive, Programming and Operating Manual, part no. 91703-93001.

HP Distributed System SCE-4 Satellite Communication Executive, Programming and Operating Manual, part no. 91704-93001.

HP Distributed System SCE-5 Satellite Communication Executive, Programming and Operating Manual, part no. 91705-93001.

HP RDTS Remote Data Transmission Subsystem, Programming and Operating Manual, part no. 91780-93001.

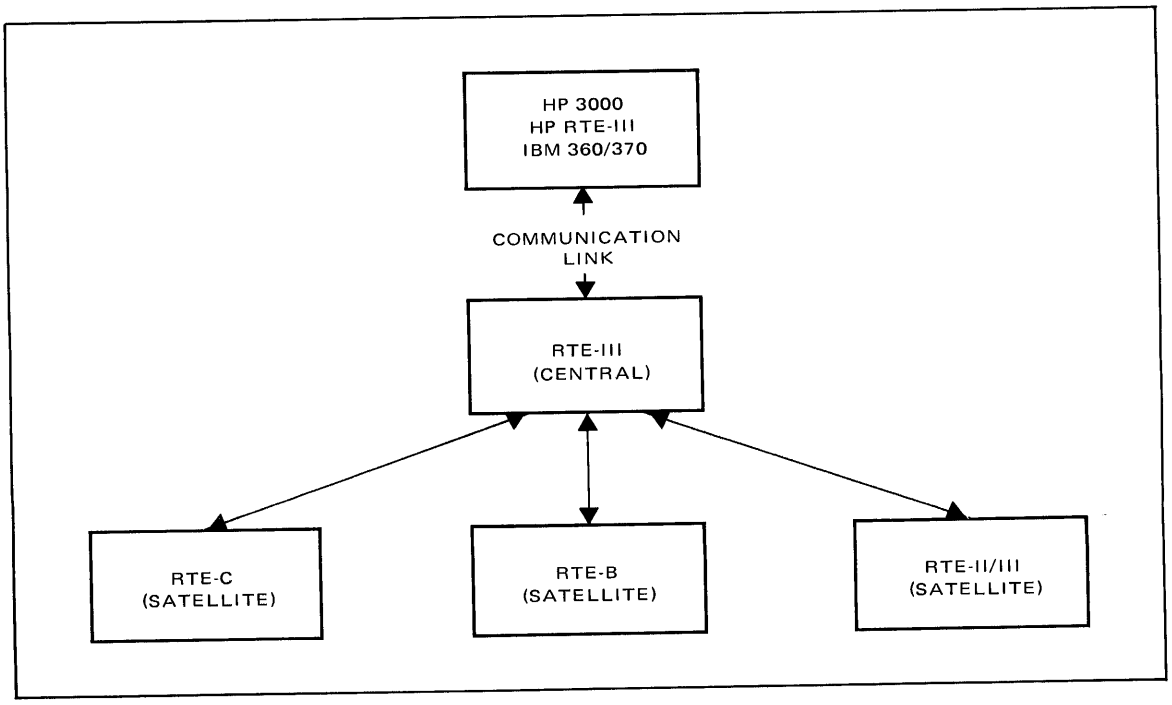


Figure 2-6. Distributed System Environment

ON LINE PROGRAM DEVELOPMENT

You can use the RTE-III operating system to extend your operation into new areas without interrupting current system functions. You can code, edit, compile or assemble, test, debug, or load new programs at the same time the system is executing existing programs.

EDITR accepts a source file from you, places it into a working area, and transfers control to you for the editing process.

LOADR accepts relocatable code from you. You may link a relocatable program file produced by the compilers or by the assembler to one or more library files using the loader. You may relocate (load) the resultant program into the RTE-III system. You may use the loader to replace programs in the system, and to obtain a list of program ID segments and general information.

Using the File Manager capability of Batch-Spool Monitor, source code, object code, and memory-image code can be saved in named files.

Figure 2-7 illustrates one of the paths that you may take to develop a program on line. The path starts at a terminal where the program is created as a file using the File Manager. A description of the program development flow follows:

1. RU,EDITR — Execute the Interactive Editor to perform edit operations on the file.

2. /ECSOURC — When the editing is completed, create a file and store it as a source file named SOURC.

3. :MS,SOURC — Move the file SOURC to the logical source (LS) track area.

4. *LG,1
   *FTN4,2,99 — Compile the program in file SOURC; the resultant code is placed in the LG track area.

NOTE

The following step, 5, is necessary only if your file is to be saved as a relocatable file. You can then use Step 6 to move the file to the LG track area at some later time.

5. :SA,RELOC — Store the LG track area contents as a relocatable file named RELOC.

6. :MR,RELOC — Move the relocatable file, RELOC, to the LG track area.

7. *RU,LOADR,99 — Execute the loader to load the program as memory image code.

8. :SP,PROGX — Save the memory image code as a program named PROGX in a type 6 file.

9. :RP,PROGX — Restore the program PROGX which was saved with the SP command.

10. *RU,PROGX — Schedule the program PROGX for execution.

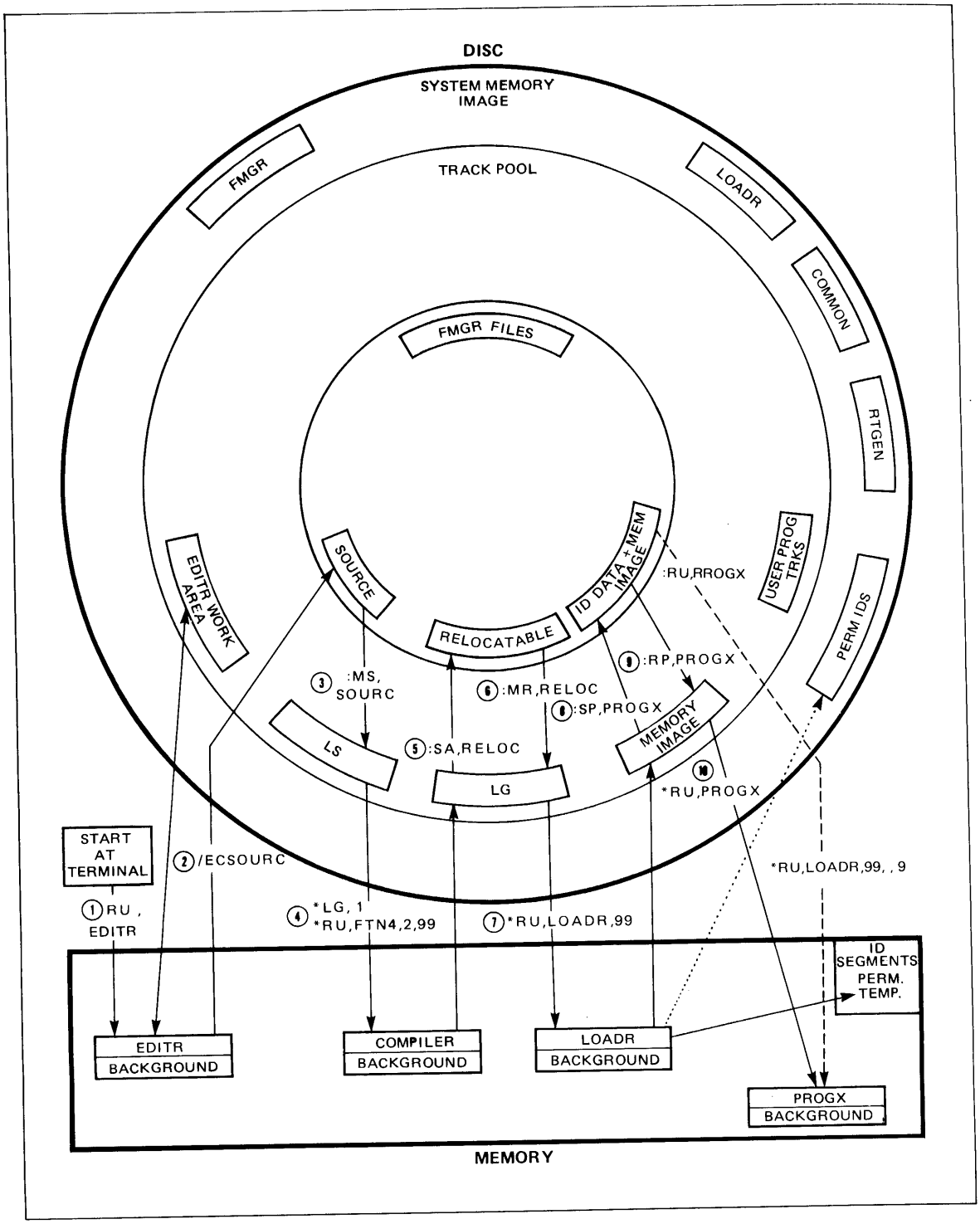


Figure 2-7. On Line Program Development Flow

MEMORY MAPPING

SECTION III

The RTE-III system requires at least 32K words of main memory. At your option, the amount of main memory can be expanded up to 256K words. This additional memory space can be divided into partitions (up to 64). Utilizing this maximum, as many as 64 of your disc resident programs can occupy main memory at a time, one in each partition. This reduces the amount of swapping necessary between main memory and disc storage.

RTE-III includes a dynamic memory mapping system which provides access to memory space greater than the 32K word limitation imposed by the 15-bit address length.

The total amount of memory available to the system (up to 256K words) is called "physical" memory. The 32K words that can be addressed at any given time is called "logical" memory. Logical memory is made up of 32 pages of physical memory. Each page is 1,024 words (1K words) in length. The 32 pages of physical memory that make up the logical memory address space need not be contiguous; they can exist anywhere within the physical memory space available.

Access to logical memory is accomplished using memory maps and the dynamic mapping system. Four maps are provided:

- System Map
- User Map
- Port A Map
- Port B Map

Only one map is in control (enabled) at one time. Each map consists of 32 registers. Each register contains the address of one page (1K words) of physical memory. That is, the enabled map describes the 32K words of logical memory address space currently accessible. A 10-bit page address in the enabled map is combined with a 10-bit relative word address resulting in a 20-bit address pointing to one word within a page.

SYSTEM MAP

The System Map is created by RTE-III during initialization of the system and remains unchanged during system operation. It resumes control whenever an interrupt occurs. The System Map describes the logical memory space which includes:

- RTE-III System
- System Base Page
- Memory Resident Library
- Common or SSGA (optional)
- System Available Memory (system work space)

USER MAP

The User Map is loaded with the appropriate set of page addresses each time a memory resident or disc resident program is dispatched for execution. Thus, the User Map is dynamic.

Memory resident programs share a single set of page addresses which describe:

- RTE-III System
- System Base Page
- Memory Resident Library
- Common or SSGA (if used by program)
- Memory Resident Program Area

Each disc resident program has its own set of page addresses which describe:

- RTE-III System
- Memory Resident Library
- Common or SSGA (if used by program)
- Program
- Program Base Page/System Communication Area

PORT MAPS

The Port A Map or Port B Map is automatically enabled each time a data transfer occurs on a Dual Channel Port Controller (DCPC) channel. The Port A Map describes the buffer space of the calling program each time a transfer occurs on DCPC channel one. The Port B Map describes the buffer space each time a transfer occurs on DCPC channel two.

Figure 3-1 illustrates the scheme used for dynamic memory mapping.

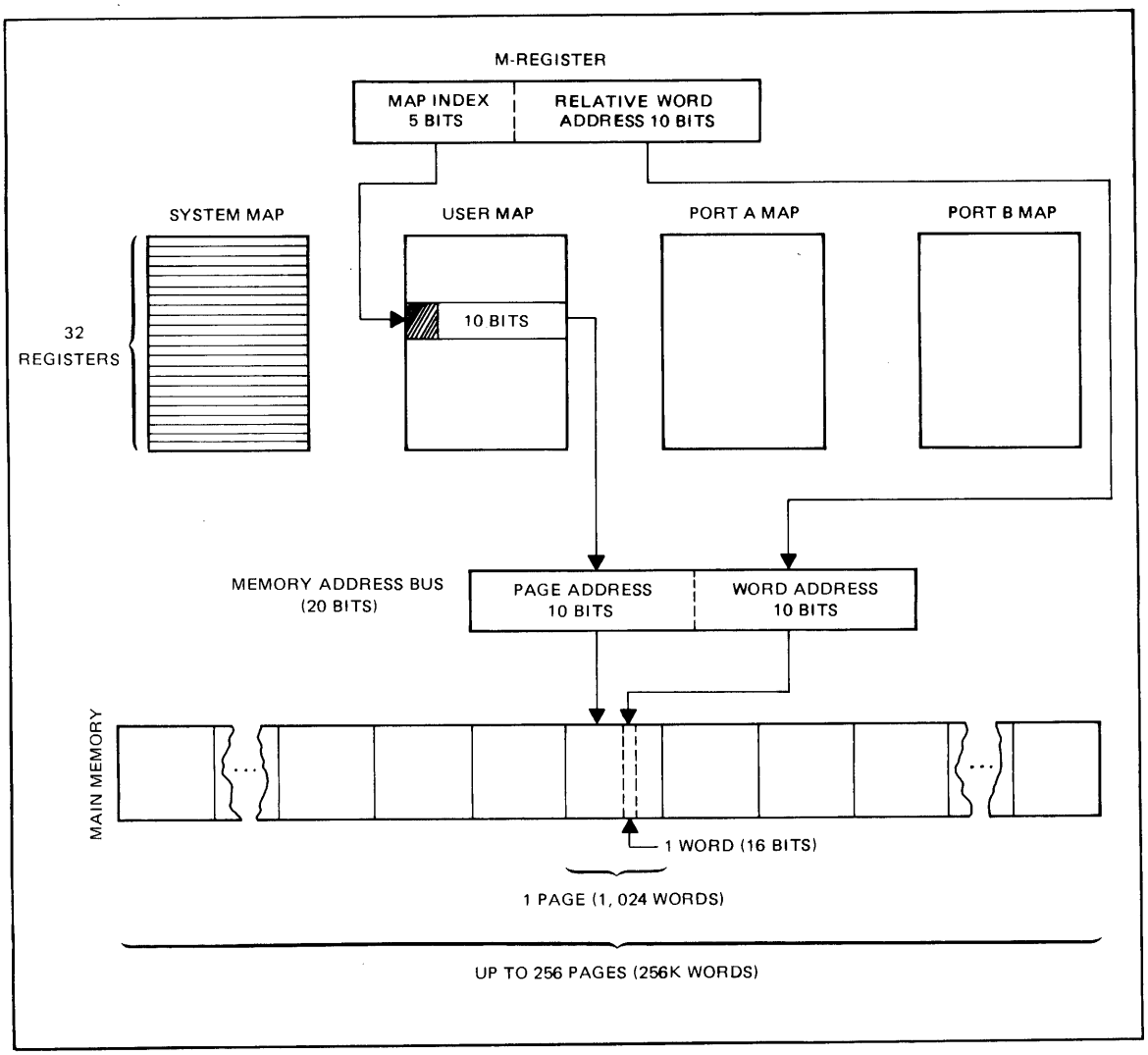


Figure 3-1. Dynamic Mapping Scheme

Figures 3-2 and 3-3 illustrate physical memory and logical memory configurations respectively.

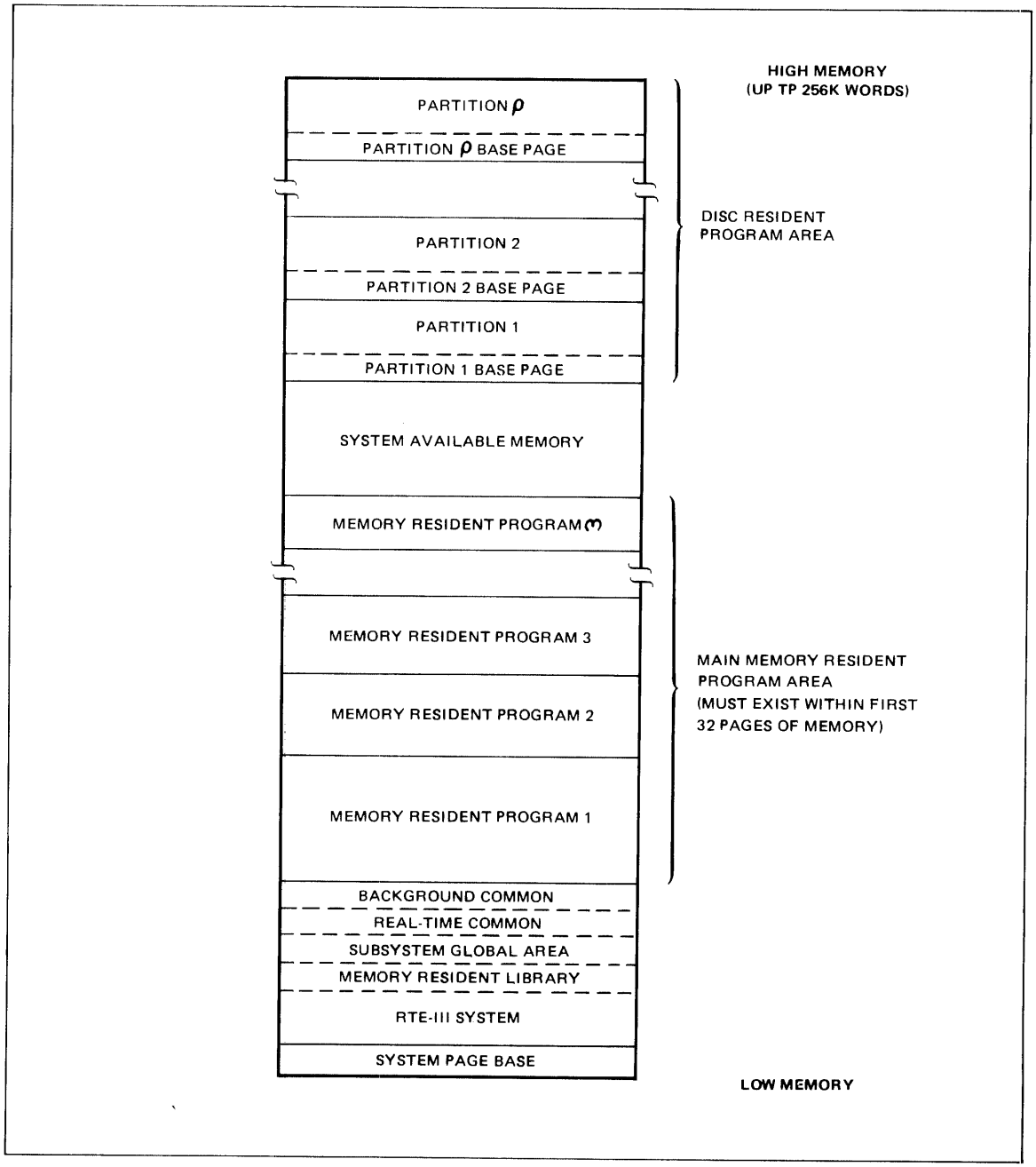


Figure 3-2. Physical Memory Configuration

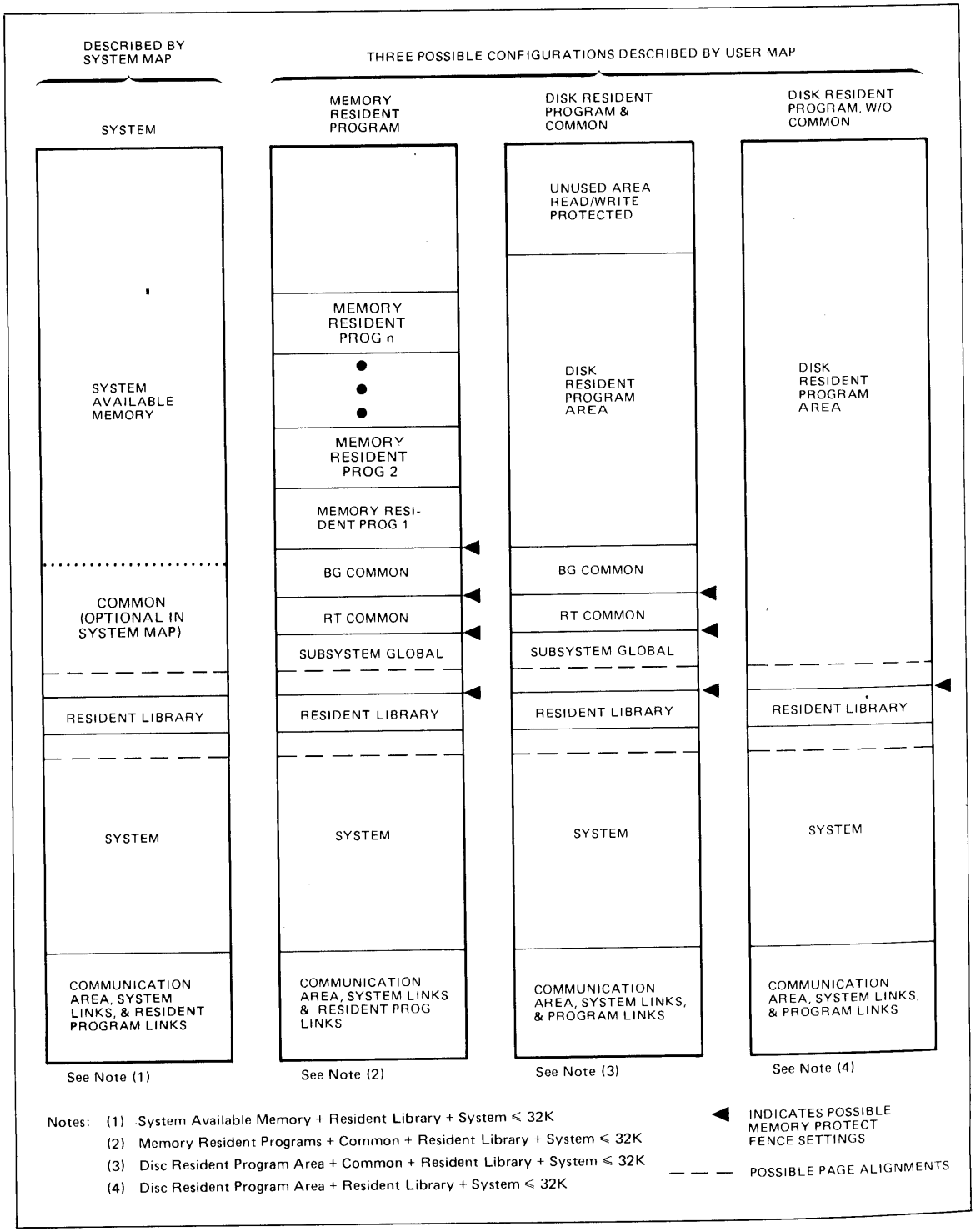


Figure 3-3. Logical Memory Configurations

RTE-III COMMANDS

SECTION IV

COMMAND STRUCTURE

Interaction between you and RTE-III is accomplished through operator commands which you enter from a terminal keyboard or through calls from your programs to the executive (EXEC) module.

OPERATOR COMMANDS

The operator commands are the interactive communication link between you and the system. These commands can be entered through a keyboard device to:

- Turn a program on or off.
- Suspend or restart a program.
- Schedule a program for execution.
- Change program priority.
- Purge a temporary program (background disc resident).
- Check status of a partition, program, or I/O device.
- Set up a source file.
- Allocate an LG track area.
- Declare an I/O device up or down.
- Dynamically alter logical unit number assignments for I/O.
- Dynamically alter I/O device buffering assignment.
- Examine or dynamically alter I/O device time out value.
- Release disc tracks assigned to dormant programs.
- Initialize the real-time clock.
- Obtain the current time from the real-time clock.

Operator commands consist of a two-character command word that may be followed by descriptive parameters. Upon entry, the command is examined for validity. If the command is legally specified, it is executed.

You must inform RTE-III that you wish to enter an operator command. This is accomplished by pressing any key on the console keyboard. RTE-III will acknowledge your signal by displaying an asterisk (*) on the console. Following the display of this system prompt, you may enter any legal command.

Table 4-1 is a list of RTE-III operator commands.

Table 4-1. RTE-III Operator Commands

COMMAND	FUNCTION
AB	Abort. Terminate the current batch job.
BL	Buffer Limit. Examine or modify the current I/O buffer limits.
BR	Break. Set the attention flag in the program ID segment to allow the program to respond to the flag if desired.
DN	Down Device. Declare an I/O device unavailable to RTE-III.
EQ	1. Equipment Status. Obtain a description and status report of an I/O device. 2. Equipment Buffering. Change the buffer designation of an I/O device.
FL	Flush Buffer. Eliminate the output from an I/O device buffer. This command is for use with Multiple Terminal Monitor (MTM) only and can be legally entered only from a terminal other than the system console.
GO	Go Program. Reschedule a program previously suspended by the Suspend (SS) operator command or equivalent EXEC call.
IT	Initialize Time. Set program execution time value.
LG	LG Track Control. Allocate or release disc tracks for an LG operation.
LS	Logical Source. Designate source file location (disc logical unit number and starting track number).
LU	1. Logical Unit Assignment. Obtain a report of the EQT entry number and device subchannel number associated with a specific logical unit number. 2. Logical Unit Reassignment. Modify a logical unit number assignment.
OF	Off Program. Terminate a program or remove a temporary disc resident program from the system.
ON	On Program. Schedule a program for execution by placing it into the time list.
PR	Priority Change. Modify the priority value of a program.
RU	Run Program. Schedule a program for execution. This command does not affect the program time list entry.
RT	Release Tracks. Release disc tracks assigned to a program.
SS	Suspend Program. Place an executing or scheduled program in the suspended list.
ST	1. Status, Program. Report the status (priority, current list, and time value) of a specified program. 2. Status, Memory. Report the name and partition number of the currently executing program. 3. Status, Partition. Report the name of the program currently occupying a specified partition.
TI	Time Request. Report the current year, day, and time of day from the real-time clock.
TM	Time Set. Initialize or change the time value in the real-time clock.
TO	Time Out Device. Report or change the time out value for an I/O device.
UP	Declare an I/O device available to RTE-III.

PROGRAM CALLS TO EXEC

EXEC calls are the communication link between your executing program and RTE-III. Using calls to EXEC, your program can:

- Perform I/O operations.
- Allocate and release disc space.
- Suspend or terminate itself.
- Schedule other programs for execution.
- Set time of execution cycles.
- Obtain partition status information.
- Obtain the current time from the real-time clock.
- Load a background disc resident program segment.

When compiled, EXEC calls result in a block of words beginning with a jump-to-subroutine (JSB EXEC) instruction followed by a list of parameters defining the request. Execution of the JSB EXEC instruction results in a memory protect violation interrupt and a transfer of control to the EXEC module. EXEC examines the parameter list and, if the call is valid (legally specified), initiates processing of the request.

In FORTRAN, calls to EXEC are coded as CALL statements or as FUNCTION calls if the A- and B-register contents are to be checked.

In ALGOL, these calls are declared as CODE procedures with parameters called by name.

In Assembly Language, calls are coded as JSB EXEC instructions followed by parameter definitions.

Table 4-2 is a list of RTE-III EXEC calls.

Table 4-2. RTE-III EXEC Calls

CALL	REQUEST CODE	FUNCTION
I/O Read	1	Transfer information from an external I/O device.
I/O Write	2	Transfer information to an external I/O device.
I/O Control	3	Pass control information such as rewind (magnetic tape) or form feed (line printer) to the I/O device.
Class I/O Read	17	Initiate the transfer of information without wait from an external non-disc I/O device or program which may be shared with other programs.
Class I/O Write	18	Initiate the transfer of information without wait to an external non-disc I/O device or program which may be shared by other programs.
Class I/O Control	19	Pass control information such as rewind (magnetic tape) or form feed (line printer) to the I/O device in a Class I/O environment.
Class I/O Write/Read	20	Initiate the transfer of information without wait to, then from (write, then read) an external non-disc I/O device or program which may be shared with other programs.
Class I/O Get	21	Complete the data transfer initiated by a Class I/O Read, Write, or Write/Read call.
I/O Status	13	Obtain I/O device status information from the device EQT entry.
Disc Track Allocation:		
1. Program	4	Allocate and assign a specific number of contiguous disc tracks to the calling program.
2. Global	15	Allocate a specific number of contiguous disc tracks which will be available to any program.
Disc Track Release:		
1. Program	5	Release some or all disc tracks previously assigned to the calling program.
2. Global	16	Release some or all disc tracks previously allocated as global tracks.
Program Completion	6	Terminate execution of calling program or a specified program.
Program Suspend	7	Suspend the calling program from execution.
Program Segment Load	8	Load calling program segment into background partition and transfer control to segment entry point.

Table 4-2. RTE-III EXEC Calls (Continued)

CALL	REQUEST CODE	FUNCTION
Program Schedule: 1. Immediate with wait	9	Schedule a specified program for execution immediately (wait for completion).
2. Immediate without wait	10	Schedule a specified program for execution immediately (no wait for completion).
3. Queue with wait	23	Schedule a specified program for execution (wait for completion). If the program to be scheduled is not available (not dormant) the calling program is placed in queue until the call can be processed.
4. Queue without wait	24	Schedule a specified program for execution (no wait for completion). If the program to be scheduled is not available (not dormant) the calling program is placed in queue until the call can be processed.
Time Request	11	Obtain current time from the real-time clock.
Timed Execution	12	Schedule the calling program for execution either after an initial offset time value or at a specified time.
Program Swapping Control	22	If the generated system allows it, the calling program can lock itself into main memory so that it will not be swapped to the disc in favor of a program with higher priority.
Partition Status	25	Obtain status information about a specified partition number.
Resource Management	*--	Allow cooperating programs to utilize the same system resource without interleaving I/O.
Logical Unit Lock	*--	Allow up to 31 programs the exclusive use of an I/O device.

*This is not an EXEC call but is a library call. It is included here for your convenience.

BATCH-SPOOL MONITOR

SECTION V

The Batch-Spool Monitor operates under supervision of the RTE-III Operating System to provide control of files, programs, batch job processing, and input/output spooling. These capabilities are provided through two separate but related subsystems — the File Management Package and the Spool Monitor. Table 5-1 lists the components of the Batch-Spool Monitor.

Table 5-1. Batch-Spool Monitor Components

BATCH-SPOOL MONITOR

1. File Management Package
   a. FMGR Program (FMGR Operator Commands)
   b. FMP Subroutine Library (FMP Subroutine Calls)
   c. *D.RTR Program (Directory Maintenance)
2. Spool Monitor
   a. GASP Program (GASP Operator Commands)
   b. SMP Program (SMP Calls)
   c. JOB Program (Job Input Spooling)
   d. *SPOUT Program (Output Spooling)
   e. *DVS43 Program (Spool Driver)
   f. *EXTND Program (File Extender)

*You cannot directly request these programs.

FILE MANAGEMENT PACKAGE (FMP)

The File Management Package handles FMGR operator commands which are used to create files, manipulate files and cartridges, and control batch jobs. It also links program calls to FMP library routines which control file access and maintenance.

FMGR PROGRAM

The FMGR program is used to manage files and to control batch processing. You schedule FMGR for execution using either the RTE-III RUN or ON command. After displaying a prompt character, the program accepts FMGR operator commands from you. FMGR is also called by the JOB program for batch processing operations. Table 5-2 is a list of FMGR commands. The command list is organized functionally.

Table 5-2. FMGR Commands

FMGR Operation

??	Request error code explanation.
:EX	Terminate FMGR and return to RTE control.
:LL	Change logical unit number of list output device.
:LO	Change logical unit number of log output device.
:SV	Change severity code.
:TE	Print message to operator at system console.
:AN	Send message to job list device.

File Creation and Manipulation

:CR	Create a file (disc or non-disc).
:PU	Purge a file.
:ST	Store data in device or file (create file).
:DU	Dump data to device or created file.
:LI	List file contents on list device.
:CN	Control non-disc file.
:RN	Rename disc file.

Program File Creation and Manipulation

:MS	Move source program to logical source area (LS).
:LS	Set or clear pointer to a logical source area (LS).
:LG	Allocate the LG track area.
:MR	Move relocatable program to LG track area.
:SA	Save LG or LS area as file (create file).
:SP	Save memory-image file for execution.
:RP	Restore memory-image file for execution.
:RU	Execute program, program file, or procedure file (transfer file).
:OF	Remove program and ID segment from memory.
:RT	Release disc tracks assigned to program.

Procedure File Manipulation

:TR	Transfer control to procedure file (transfer file) or logical device.
:PA	Pause and send message to log device.
:DP	Display parameter values.
:SE	Define global parameters.
:CA	Calculate values of global parameters.
:IF	Test and Branch on parameter values.

Batch Job Control

:JO	Initiate job.
:EO	Indicate end-of-job.
:AB	Terminate job.
:TL	Set time limit within job.
:LU	Switch logical units within job.
:CS	Change spool logical units.

Table 5-2. FMGR Commands (Continued)

FMP Cartridge Manipulation	
:MC	Mount cartridge.
:IN	Initialize cartridge.
:CL	List cartridge directory.
:DL	List file directories.
:DC	Dismount cartridge.
:PK	Pack cartridge.
:CO	Copy all files from cartridge to cartridge.

FMP CALLS

The FMP subroutine calls are issued from your program to access, maintain, and manipulate files. Table 5-3 is a list of FMP calls and their functions.

Table 5-3. FMP Subroutine Calls

CALL	FUNCTION
APOSN	Position a file to a specific record.
CLOSE	Close a file.
CREAT	Create a file.
FCONT	File control.
FSTAT	File directory status request.
IDCBS	Data Control Block (DCB) buffer size request.
LOCF	File status request.
NAMF	Rename a file.
OPEN	Open a file.
POSNT	Position a file to a record relative to the previous record.
POST	Data Control Block (DCB) buffer flush (write to a file).
PURGE	Purge (delete) a file.
READF	Transfer a record from a file to a buffer.
RWNDF	Reset position pointer to beginning of file.
WRITF	Transfer a record from a buffer to a file.

SPOOL MONITOR

The Spool Monitor is an optional segment of the Batch-Spool Monitor. If selected, it operates in conjunction with the File Management Package (FMP). Use of the Spool Monitor adds spooling capabilities to the batch job processing operation.

A batch job is a set of FMGR commands which fit together to perform tasks. The set of commands must begin with a JO command and end with an EO command.

Batch job processing without spooling is a simple "input/process/output" procedure. FMGR reads the job commands directly from the input device and transfers processed data directly to an output device. Figure 5-1 illustrates this type of batch job processing.

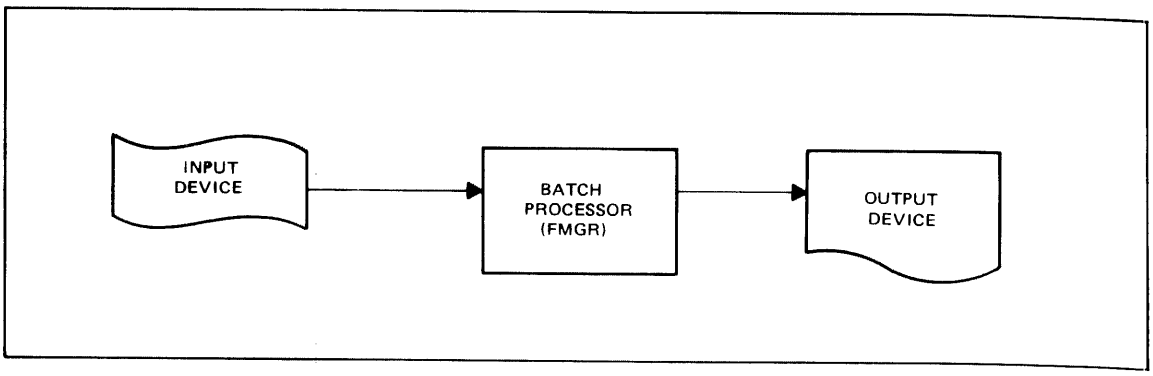


Figure 5-1. Batch Job Processing without Spooling

Spooling is the buffering of batch job input and output on disc files called spool files. The addition of spooling capabilities to batch job processing increases the amount of data that can be processed within a specific time period. Slow input and output devices do not hinder the processing of data because the data is "spooled" into disc files and, subsequently, out of these spool files. Figure 5-2 illustrates batch job processing with spooling.

Functions provided by the Spool Monitor include:

- Opening and closing spool files. Upon close of a spool file, the contents of the file are written to a user selected non-disc device for output.
- Maintaining a record of the current status of all jobs and spool files in the system.
- Translating non-disc device references in program I/O calls into references to disc files known as spool files.

The process of spooled input is called inspooling while that of spooled output is called outspooling.

Inspooling is the process of accepting a job from a serial input device and placing that job in a disc file called a spool file. The job is executed from the spool file according to its priority. Resultant output is placed in a spool file.

Outspooling is automatic for jobs entered via the JOB program, no interaction with the system is required to complete a job. Figure 5-3 is a simplified diagram of batch job flow using the Spool Monitor. Figure 5-4 shows the events that occur during the spooling process.

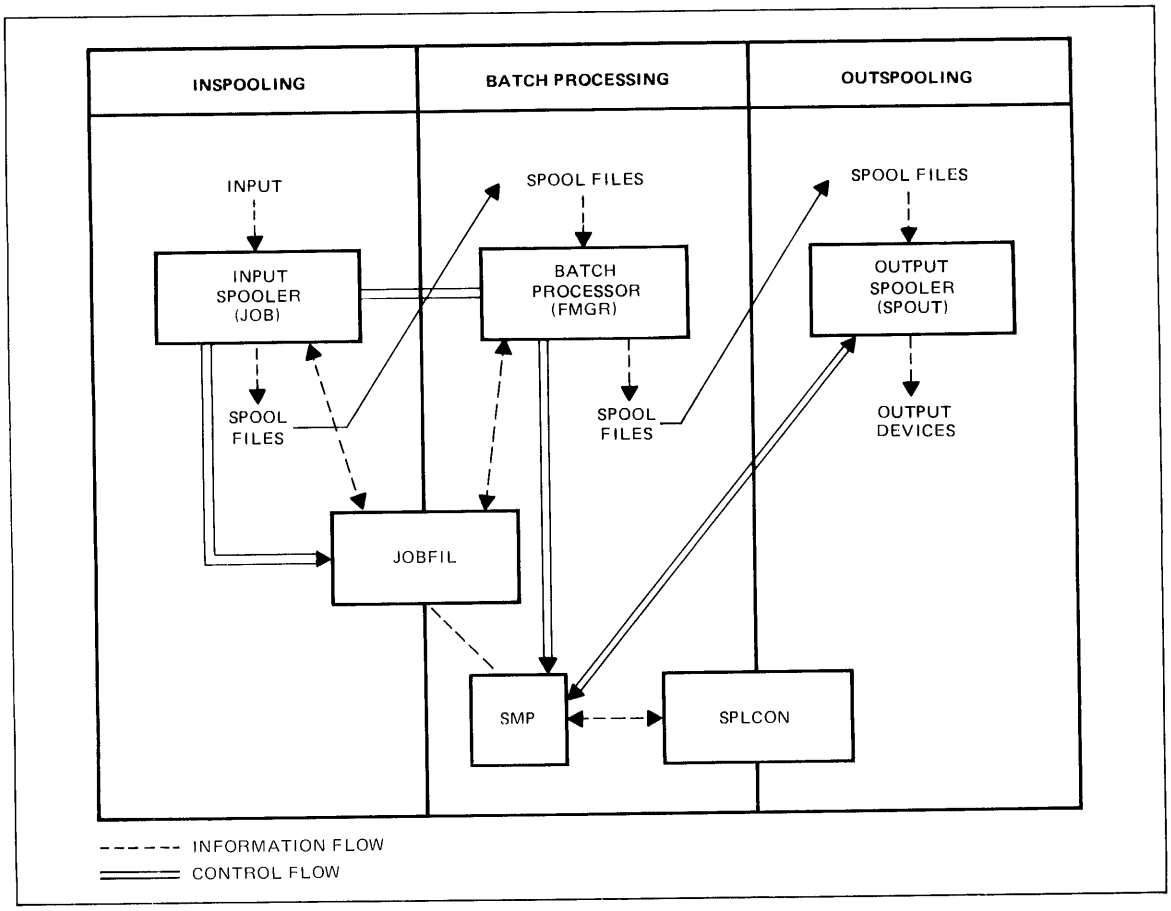


Figure 5-2. Batch Job Processing with Spooling

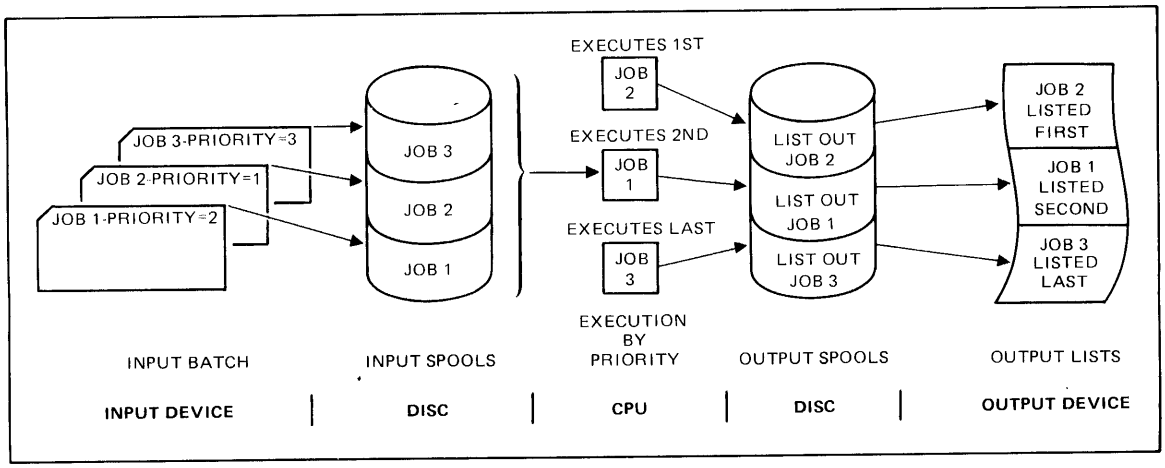


Figure 5-3. Batch Job Flow with Spooling

Inspooling (JOB Program)	Processing (FMGR Program)	Outspooling (SPOUT Program)
JOB is started: Reads input . . . JOB finishes reading a job: Updates JOBFIL queue, schedules FMGR: JOB starts reading next job. . . . JOB finishes reading a job. . . . . Job continues operating until there is no more input to read.	FMGR starts: Looks at JOBFIL queue and finds job to process. Sets up read spool. Sets write spools. FMGR continues to process this job. . . . . FMGR finishes job. Closes spool files and gets next job. . FMGR continues processing until there are no more jobs left in the JOBFIL queue.	SPOUT is scheduled when first write spool is set up. Starts transferring data to devices as soon as a file is queued and output exists. . . . .

Figure 5-4. Spooling Event Chart

GASP PROGRAM

The GASP program permits you to interactively control I/O spooling operations. You schedule GASP for execution using either the RTE-III command RUN or ON. After displaying a prompt character, GASP accepts entry of commands which control the spool environment. The following commands (Table 5-4) are supported by the GASP program:

Table 5-4. GASP Operator Commands

COMMAND	FUNCTION
AB	Abort a job before it is run.
CJ	Change job status (job priority, or hold/release state).
CS	Change spool status (spool priority, or hold/release state).
DA	Deallocate all spool files and spool control files.
DJ	Display job status.
DS	Display Spool Status.
EX	Exit (terminate) GASP program.
KS	Kill (purge) spool file in outspooling queue.
RS	Restart outspooling operation from the beginning.
SD	Shut down Batch-Spool Monitor.
SU	Start up Batch-Spool Monitor following a shut down.
??	Request an explanation of an error (expand message).

SMP CALLS

The Spool Monitor program (SMP) provides a callable subroutine named SPOPN which is used to open a spool file. Once a spool file is open, SMP calls are available to control outspooling from user programs. These calls are used to:

- Save a file at the end of an operation.
- Purge a file at the end of an operation.
- Pass the spool file to the outspooling queue immediately.
- Close the spool file and pass it to the outspooling queue.
- Change the outspooling logical unit number, or priority.
- Set the buffering flag in the outspooling EQT entry.
- Clear the buffering flag in the outspooling EQT entry.
- Obtain spool file position information; that is, the position of the record pointer.
- Change the spool file starting position.

INDEX

READER COMMENT SHEET

RTE-III General Information Manual

92060-90009 **FEB 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. 92060-90009
Printed in U.S.A. 2/76

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
11000 Wolfe Road, Cupertino, California 95014