RTE-IVB

MULTI-TERMINAL OPERATION

USERS GUIDE

GRUMMAN

AT520G4-000

FEBRUARY 1985

TABLE OF CONTENTS

<u>APPENDICES</u>

SECTION		PAGE

SECTION I

INTRODUCTION

1-1 INTRODUCTION.

1-2 The objective of this manual is to describe the operation and capabilities of the RTE Operating System.

1-3 RTE Operating System henceforth referred to as RTE, represents a third generation, user oriented, operating system. This system is a derivative of the Hewlett- Packard RTE-IVB Operating System. In addition to RTE, the Test System has the following key features:

a. Microprocessor controlled, high performance universal switching system

b. Resident Interactive ATLAS Compiler

c. Simple operation and programming

d. IEEE-488 Bus Standard (optional)

e. Modular hardware and software systems

f. Compatible with HITS, LOGOS and LASAR Automatic Test Generator

g. Automatic system self-test assures the functional integrity of the test station.

1-4 RTE HARDWARE REQUIREMENTS.

1-5 The RTE Software is designed around four major components, as follows:

a. Computer subsystem

b. Keyboard/CRT Terminal

c. Disc Drive

d. Line Printer.

1-6 COMPUTER SUBSYSTEM.

1-7 The functions performed by the computer subsystem include the following:

a. Provides the operator/machine interface for test station operation

b. Controls the various signal sources and sensor instruments

c. Provides the capability of entering, editing, and compiling application programs

d. Controls and monitors test program execution.

1-8 CONTROL COMPUTER. The control computer (HP21 MX) is the heart of the computer subsystem. It performs all of the control functions relating to test station operations. Some of the characteristics of this computer are:

a. 16 bit word size and 128 standard instructions including 80 instructions to provide upward capability with the HP2100 series computer

b. 4 general purpose registers, two of which may be used as index registers

c. Fully microprogrammed processor, including all arithmetic functions, input/output, and operator panel control

d. Initial binary loader is ROM resident and callable by pushbutton switch on the operator panel

e. Continuous power fail monitoring with automatic restart capability.

1-9 KEYBOARD/CRT TERMINAL.

1-10 The Keyboard/CRT Terminal serves as the test station control device. It provides the necessary man/machine interface for operator communications with the test station. Typical applications include:

a. Entering of control commands.

b. Entering and editing of application programs on-line.

c. Display of system messages and operator instructions.

1-11 DISC DRIVE.

1-12 The Disc Drive is used for storage of the operating system and application programs. It is a random access device. A detailed description of the disc is given in Appendix B.

1-13 LINE PRINTER.

1-14 The Line Printer can be used to provide hard copies of application program listings, failure data during test program execution, as well as output generated by the system or system utilities.

1-15 MAJOR SOFTWARE AREAS.

1-16 The principal function of an Operating System is to manage the use of system resources required by user's programs. These programs compete for use of system resources such as central processor (CPU) time, main memory, disc storage, I/O devices, and system library software. The user assigns a priority to each of his programs. The Operating System uses this value to determine which program, among those competing for specific resource, will actually be given access to that resource. The program with the highest priority is granted access to the resource.

1-17 RTE-IVB is a disc based operating system that provides the supervisory functions necessary to coordinate requests for, and allocation of, system services and resources. Being a real-time system, RTE-IVB processes all decision and scheduling tasks internally unless overridden by user intervention. User requests for system action can be made by a "call" from within a program or interactively via an operator command.

As the major control element within the operating environment, RTE-IVB provides the user with various services and automatically handles the machine-related functions associated with each service. The major services provided by RTE-IVB are briefly summarized below:

- Executive Communication scheme that provides a communication link between user-written programs and system services.

- Segmentation technique that allows a large program to be separated into a main program and related segments, thereby allowing it to execute in a memory partition smaller than its total size.

- Resource Management capabilities that allow cooperating user-written programs to share system resources (files, I/O devices, etc.).

- I/O scheme which allows a program to continue executing while its own I/O requests are being processed.

- Program execution control that features multiprogramming (allows several programs to be active concurrently) and time-slicing (prevents compute intensive programs from dominating the CPU).

- Partitioned memory technique that takes advantage of the hardware Dynamic Mapping System (DMS) to provide access to 2048k bytes of physical memory.

- Extended Memory Area (EMA) that allows user-written programs to access large data arrays; the size of the arrays being limited only by the size of physical memory.

- Operator interface that provides the user with the ability to control system action via operator commands.

1-18 RTE-IVB is a multiprogramming system that allows several programs to be active concurrently; each program executes during the unused central processor time of the others. Scheduling/dispatching modules in RTE-IVB decide when to execute programs that are simultaneously requesting system services and/or resources. The scheduling module places programs into a scheduled list in order of their priority (the highest priority program at the head of the list) and the dispatching module initiates the execution of the highest priority program. Programs with the same priority are scheduled on a first-come-first-serve basis. When the executing program completes, is terminated, or is suspended, it is removed from the scheduled list and the dispatching module transfers control to the next program with the highest priority. Note that the next program to be executed could have the same priority as the program that was just removed from the list.

The scheduled list can be logically divided into two areas by placing a time-slicing boundary at a priority level. Programs with priorities that place them above the boundary (higher priority, lower numerically) are executed in the linear fashion described above.

Programs with priorities that place them below the boundary (lower priority, higher numerically) are executed in a similar fashion with one exception; programs are assigned an execution interval when they are scheduled. When a program exceeds its interval, it is moved within its priority level in the scheduled list.

Each priority level below the time-slicing boundary can be considered a queue. The program at the head of each priority queue represents the next program of that priority to be executed. When the execution of the program at the head of the queue is initiated, a maximum time interval for execution (time quantum) is calculated by the operating system. The program is allowed to execute until one of the following occurs:

1. The program leaves the scheduled list (I/O suspended, memory suspended, etc.).

2. A higher priority program is ready to execute.

3. The program exceeds its time quantum.

If a program leaves the scheduled list, its time quantum is assumed exhausted. When the program is again ready to execute, it is placed at the end of the queue within its priority in the scheduled list and a new time quantum is established.

If a higher priority program causes the suspension of a time-slicing program, the remaining portion of the suspended program's time quantum is saved in its ID segment. When the suspended program is scheduled to continue executing, the saved quantum value is restored.

When a time-slicing program exceeds its time quantum, it is placed at the end of the queue within its priority in the scheduled list and control is transferred to the new head of the queue. Figure 1-1 shows a diagram of scheduling with time-slicing.

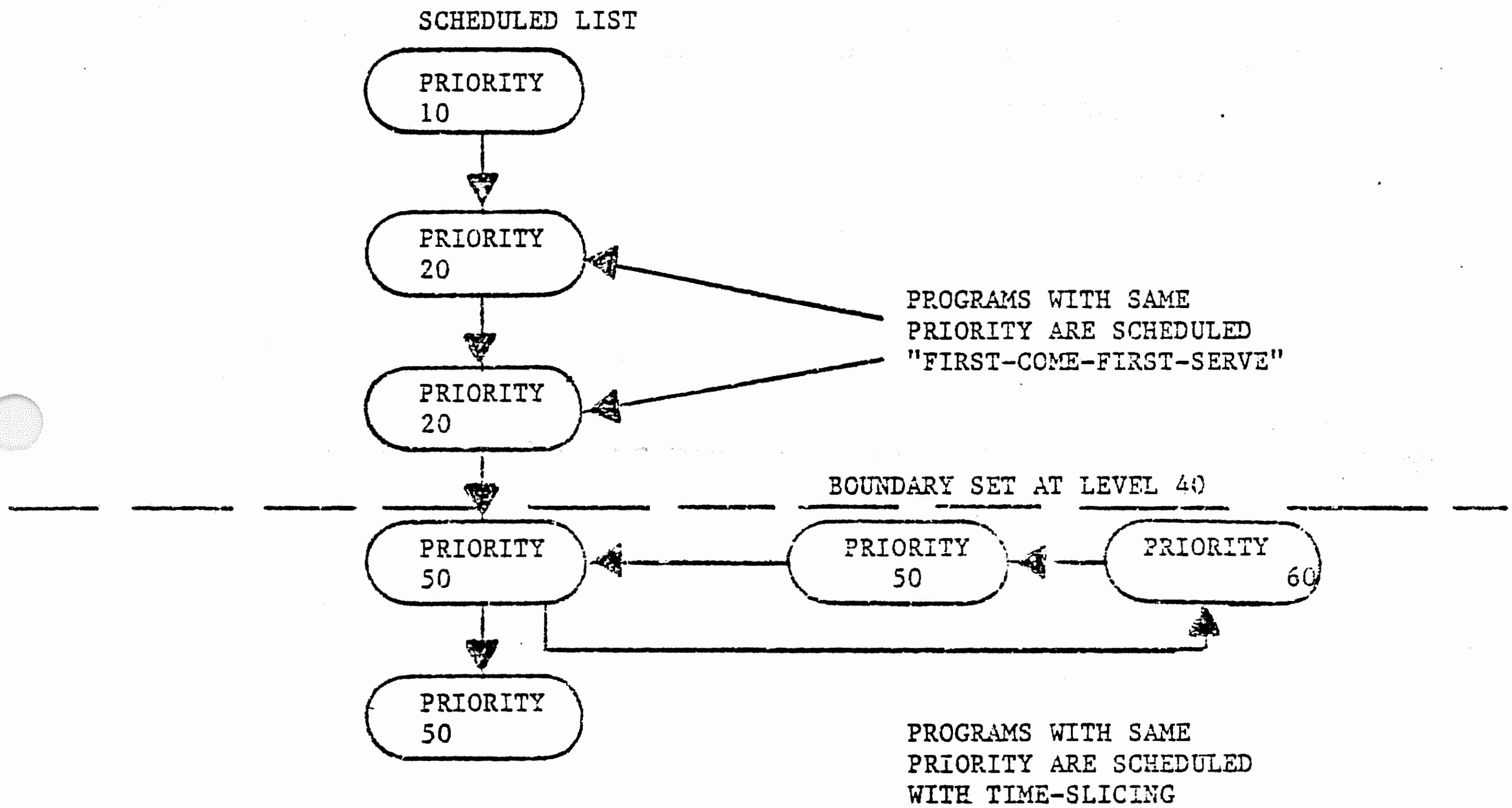


Figure 1-1. Scheduling with Time-Slicing

1-19 CONTROL OF RTE.

1-20 RTE is controlled through operator commands entered from a system console device (or auxiliary terminal). The operator controls the RTE Operating System by using a set of RTE Operator Commands. These commands are used interactively. The Operating System issues an appropriate response to each command entered. The response may be a message or simply a carriage return, line-feed. Additional operator commands are provided by the File Manager (FMGR). This is the file subsystem of RTE.

1-21 LANGUAGE SUPPORT.

1-22 RTE provides several programming languages as well as a relocatable subroutine library. The supported languages are:

a. 21MX ASSEMBLER

b. FORTRAN IV

c. ATLAS TEST LANGUAGE PROCESSOR

SECTION II

GENERAL INFORMATION

2-1 SYSTEM OVERVIEW.

2-2 The RTE-IVB operating system is written to take advantage of the hardware Dynamic Mapping System (DMS) available with the HP/1000 computers. The cooperation between the software operating system and the hardware mapping system allows access to 1024K words of physical memory.

The basic addressing space of the HP/1000 computer is 32,768 words (32K) as defined by the 15-bit address length used by the CPU. This is referred to as logical memory. The amount of memory actually installed in the computer system is referred to as physical memory. The DMS maps the 32K words of physical memory into logical memory by translating the 15-bit address through "memory maps".

MEMORY MAPS

The page pointers contained in the map registers that comprise the memory maps are loaded by software modules within the operating system. Each map is configured to represent the 32 pages of physical memory (not necessarily contiguous) that contain the tables, buffers, data, program code, etc., necessary to perform specific processing tasks. Since there are four maps, four different 32 page sections of physical memory can be described simultaneously, with only one map being enabled at a time to indicate the section of memory currently in use. The maps are altered by the operating system to reflect dynamic changes in the operating environment, i.e., when a program is scheduled to execute, a map has to be configured to describe the physical memory pages it requires (known as the program's logical address space).

The four memory maps are classified by the type of processing tasks they are associated with. The maps are comprised of the following:

- System Map - The System Map describes the logical address space associated with the RTE-IVB operating system and its base page, COMMON, Subsystem Global Area, Table Area I and II, driver partition, System Driver Area (SDA), and System Available Memory (SAM). The system map is loaded during system initialization and is changed only to map in different driver partitions. Since the RTE-IVB operating system handles all interrupt processing, the system map is automatically enabled whenever an interrupt occurs.

- User Map - The User Map is associated with each disc resident program. It is a unique set of pages that describe the logical address space containing the program's code, the program's base page, Table Area I, driver partition, and optionally, Table Area II, System Driver Area, and COMMON.

All memory resident programs use a common set of pages that define the memory occupied by the memory resident program and its base page, the Memory Resident Library, Table Area I, driver partition, COMMON, and optionally, Table Area II and System Driver Area.

Each time a new memory or disc resident program is dispatched, the system reloads the User Map with the appropriate set of pages.

- o Port Map A and Port Map B - The Port maps are associated with DCPC transfers. DCPC transfers are software assignable direct data paths between memory and a high speed peripheral device.

  This function is provided by the Dual Channel Port Controller (DCPC). There are two DCPC channels, each of which may be assigned to operate with an I/O device. Port A Map is automatically enabled when a transfer occurs on DCPC channel 1, and Port B Map is enabled when a DCPC channel 2 transfer occurs.

  DCPC transfers are accomplished by "stealing" CPU cycles instead of interrupting the CPU and transferring to an I/O service routine. Implementation by "cycle-stealing" facilitates multiprogramming since one program can be executing via the User Map while a DCPC transfer is in progress on another program's data buffer.

  The Port Maps are reloaded by the system each time a DCPC channel is assigned for an I/O request. The Port Maps will be the same as the System Map or the User Map associated with the program being serviced, depending on the type of request. Once initiated, the DCPC transfer is transparent to the user since the currently enabled map (System or User) shares the CPU with the Port Maps, i.e., during a given instruction cycle (comprised of several CPU cycles) the System or User Map is enabled alternately with the Port Map.

2-3 PHYSICAL MEMORY.

At generation time, the user plans physical memory allocations and loads the system components and drivers for the most efficient configuration. The user determines the size of System Available Memory (SAM), 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.

The following is a brief description of the Physical Memory Configuration shown in Figure 2-1.

- o System Base Page - Contains system communication area and is used by the system to define request parameters, I/O tables, scheduling lists, pointers, operating parameters, memory bounds, etc. System links and trap cells are also located on the system base page.

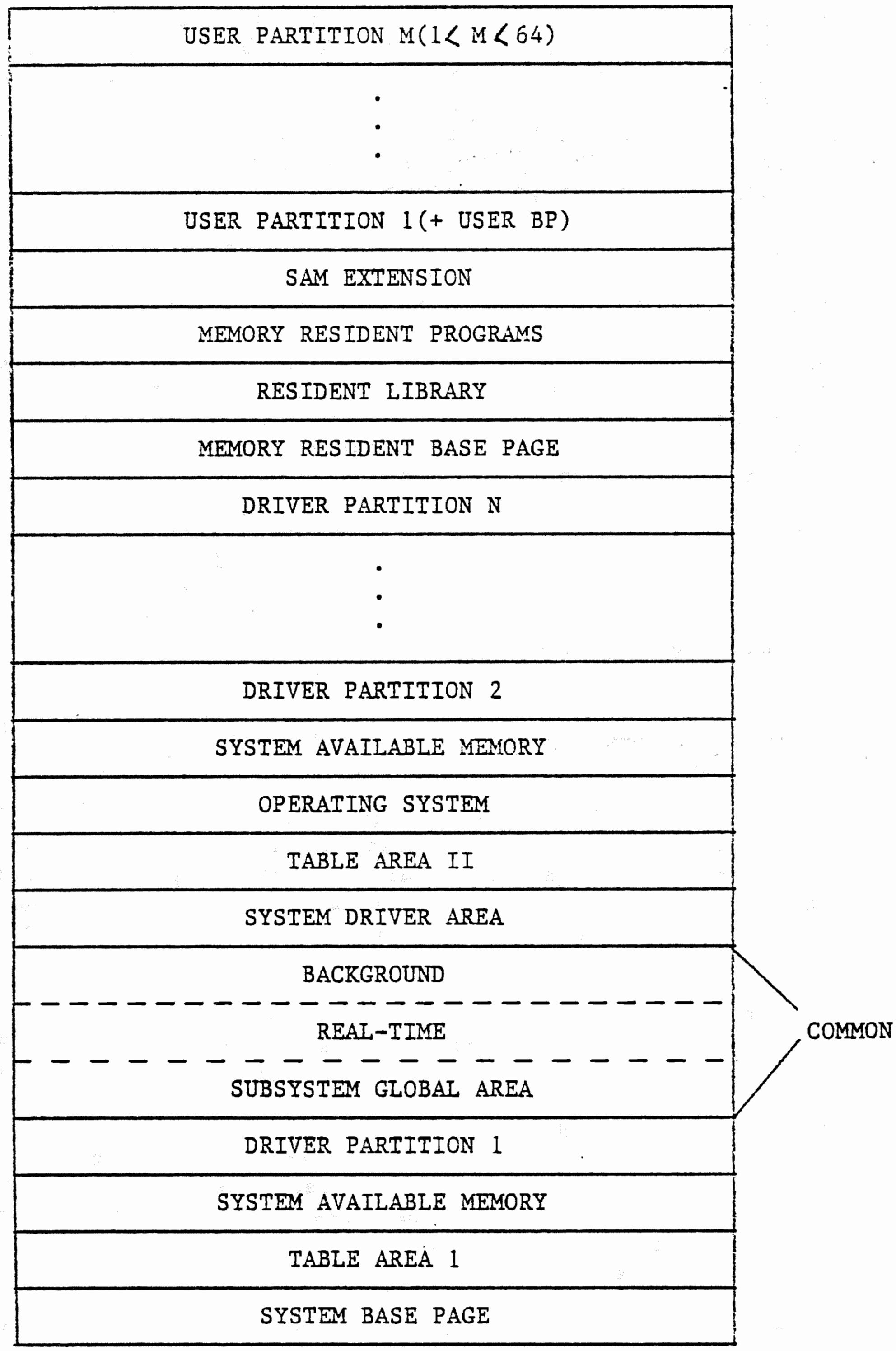


Figure 2-1. Physical Memory Allocations

Base page links for the memory resident library and memory resident programs are only in the memory resident base page and are not accessible by disc resident programs. The Table Areas, SSGA, driver 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. For all practical purposes, the memory resident programs are in a single partition separate (protected) from all other partitions.

- Table Area I - Contains the Equipment Table entries, Driver Mapping Table, Device Reference Table, Interrupt Table, the Disc Track Map Table, some system and HP subsystem entry points.

- Driver Partition - An area, established at generation time, containing one or more drivers. All driver partitions are the same length, and only one driver partition is included in a 32K word address space at any one point in time. The minimum partition size is two pages but may be increased when the system is generated.

- System Driver Area - An area for privileged drivers, large drivers, or drivers that do their own mapping. The drivers that go into this area are specified during the EQT definition phase of system generation. The System Driver Area (SDA) is included in the logical address space of both the system and Type 2 and 3 programs. It is included in the memory resident program area (if requested) at generation time.

- System - Contains the absolute code of system modules (i.e., RTIOC, SCHED, EXEC, etc.).

- Memory Resident Library - Contains the reentrant or privileged library routines (Type 6) that are used by the memory resident programs, or which are force loaded at generation time (Type 14). It is accessible only by memory resident programs. All routines loaded into the resident library also go into the relocatable library for appending to disc resident programs that require them.

- 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.

  These system COMMON areas are not to be confused with the local COMMON area that 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 segments.

- Subsystem Global Area - Reserved for modules required at generation time.

- Memory Resident Programs - This area contains all Type 1 programs that were relocated during generation.

- Table Area II - Contains the Keyword Table, ID segments, ID Segment Extensions, Class Table, RN Table, Batch LU Switch Table, Memory Resident Map, and a number of entry points for system pointers. This area has entry points that are created by the generator.

- System Available Memory - This is a temporary storage area used by the system for buffering, Class I/O, reentrant I/O, and parameter string passing.

- Partition - This is an area established by the user for a disc resident program to execute in. Each partition has its own base page that describes the linkages for the program running in the partition. Up to 64 partitions are allowed, within the constraints of available physical memory.

- Memory protection is provided by a combination of the Dynamic Mapping System and the Memory Protect Fence. DMS provides protection between program partitions by not allowing a program to access memory locations that are not defined by its memory map. The Memory Protect Fence prevents a program from addressing memory locations below a given address within its memory map.

- Partitions are blocks of physical memory reserved for disc resident programs. Program partitions are defined during system generation and may be redefined during the reconfiguration process at system boot-up.

Programs within the RTE-IVB operating system are categorized according to where they reside (memory- or disc-resident), what type of memory partition they execute in, by the COMMON areas they have access to, and whether or not they may be duplicated.

A program's type is user-defined in the program definition statement (PROGRAM statement in FORTRAN, NAM statement in Assembly Language) or is assigned by the user when the program is loaded. If no type parameter is specified, a default value is determined and assigned by the loader.

2-4 REAL-TIME ASSEMBLER.

2-5 The real-time assembler accepts storage programs in assembler language and outputs a relocatable binary program.

2-6 REAL-TIME FORTRAN IV.

2-7 Real-time Fortran IV compiles source programs written in Fortran IV, with extended precision, arithmetic, and bit manipulation via logical expressions.

2-8 REAL-TIME ATLAS.

2-9 The Grumman ATLAS compiler allows an operator to edit or enter a program, statement-by-statement, and execute it immediately. Since no intermediate language is used, lengthy and time consuming recompiling of programs is avoided. Source and object programs can be stored on disc, listed, displayed, or executed directly.

2-10 The ATLAS language elements include all of the necessary computational statements for performing full arithmetic, looping, disc accessing and branching operations required for test programming. A full complement of instrument statements is provided to control digital, analog, switching and graphical functions. ATLAS operator and control commands for edit, storage, execution, etc. are standard features.

2-11 REAL-TIME INTERACTIVE EDITOR.

2-12 Refer to Appendix A for a detailed description of RTE editing capabilities.

2-13 REAL-TIME RELOCATING LOADER.

2-14 The real-time relocating loader provides on-line loading of system and user programs written in Fortran and Assembler language that have been compiled or assembled to relocatable object code. The loader will then generate absolute executable code from the relocatable modules.

2-15 PROGRAM SCHEDULING (SCHED).

2-16 Scheduling of all programs is done by the scheduling module SCHED, and is based on priority. Programs may be scheduled for execution by an operator request, a program request, a device interrupt, or the completion of a time interval. The RTE system can be generated such that one program is automatically scheduled each time the system is loaded from the disc. Whenever programs conflict because of simultaneous demands for execution, SCHED decides in favor of the highest priority program. Priorities are assigned by the user during RTGEN or on-line loading, and may be changed by an operator request.

2-17 The RTE system handles priorities on a completely generalized basis. The highest priority program scheduled for execution executes first. Then, if that program suspends execution (possibly for an input wait), the next highest priority scheduled program executes. This process continues down the scheduled list until a higher priority program is rescheduled.

2-18 INPUT/OUTPUT.

In the RTE-IVB operating system, centralized control and logical referencing of I/O operations effect simple, device-independent programming. All I/O and interrupt processing is controlled by the operating system with the single exception of privileged interrupts (privileged interrupts circumvent the system for faster response time).

Requests for I/O services are made by EXEC routine calls coded into the calling program. The EXEC calls specify the type of transfer (Read, Write, Control) and the desired device. I/O requests from a particular program are queued to the controller's I/O list according to the calling program's priority. Automatic buffering for write operations is provided if specified at generation.

2-19 LOGICAL UNIT NUMBERS.

Logical Unit numbers provide RTE users with the capability of logically addressing the physical devices defined by the Equipment Table. Logical Unit numbers are used by executing programs to specify which I/O device requests are to be directed to. In an I/O EXEC call, the program simply specifies an LU number and does not need to know which physical device or which I/O controller handles the transfer.

An LU is associated with an EQT entry and a subchannel. Some I/O devices have EQT entries with one subchannel designation (i.e., line printers) and are referenced by a single LU number. Other devices (disc drives and CRT terminals) have EQT entries with several subchannel designations, with an LU assignment for each subchannel. When a user makes an I/O request specifying an LU number, he can be addressing a total device (line printer) or a subsection of a device (one surface of a disc).

2-20 I/O CONTROLLER TIME-OUT.

Each I/O controller may have a time-out clock to 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 that made the I/O call would remain in I/O suspension indefinitely, waiting for the "operation done" indication from the device's controller.

2-21 FILE MANAGER (FMGR) PROGRAM(S).

2-22 The File Manager is used to manage user generated files. It responds to commands entered via the keyboard. For a discussion of File Manager Organization see Section IV. See Appendix D for the File Manager Commands.

2-23 DISC DRIVE FUNCTIONAL CHARACTERISTICS.

2-24 The RTE moving head disc handler is responsible for controlling data transmission to and from the disc drive. This handler operates under the control of the I/O control module of RTE.

2-25 The operating system subdivides the disc into logical sectors. A logical sector consists of 64 words. Since a physical sector is 128 words one physical sector contains two logical sectors. All of the bookkeeping, blocking, and address calculations are handled by the disc handler and is therefore, transparent to the user.

Refer to Appendix B for a more detailed discussion.

2-26 OPERATING PROCEDURES

Refer to Appendix C for RTE Operating Procedures.

2-27 FILE MANAGEMENT.

The File Manager controls the initializing, reading, and writing of disc logical units. All user removable discs are configured the same way. This uniformity gives the user a great deal of flexibility with respect to using different RTE installations. A given user's removable disc configured for RTE would be acceptable to any other RTE installation using the same version of the operating system.

The File Management Package (FMP) allows the user to manipulate I/O devices and files. The user interface to the FMP can be either interactive (using FMGR commands) or programmatic (using FMP calls).

The FMP library contains routines that are called from user programs and used to manipulate disc and non-disc files. Using calls to these routines, the user can create, access, purge, and obtain the status of files.

Files are classified according to the record format within the file and the type of data the system expects to find in each record. A file's type is defined when it is created and this information is placed in it's file directory entry. When a file is accessed, this information is used by FMP to determine the file's characteristics and initiate the appropriate action as specified by the type of file it is manipulating.

ATLAS and BASIC (where applicable) will automatically use those types of files which will produce the most efficient program. The user need not concern himself with file types while using ATLAS or BASIC.

A file can contain up to (2**31)-1 records and can have a total size up to 32767 X 128 blocks (1 block = 256 bytes). For files with fixed-length records, the file size is defined at creation and cannot be expanded. For files with variable-length records the base file size is defined at creation and the file is extendable as needed.

The following is a summary list of the services available to user programs via FMP calls:

- o File creation (disc file only)
- o Opening files for specific modes of access
- o Read and write access to files
- o Positioning to records within a file
- o Closing files to access
- o Purging files from the system
- o Obtain position and status information on files
- o Renaming of files
- o Obtain disc cartridge list.

SECTION III

OPERATOR REQUESTS (SYSTEM COMMANDS)

3-1 INTRODUCTION.

3-2 The operator controls an executing Real-Time Executive System by operator requests entered through the console. These operator requests can interrupt RTE to perform the functions described in Table 3-1.

3-3 COMMAND STRUCTURE.

3-4 The operator gains the attention of RTE by pressing any key on the console. When RTE responds 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 resolved by a central routine that accepts certain conventions. Command syntax is described in Table 3-2 and, with the conventions described next, must be followed exactly to satisfy system requirements.

3-5 COMMAND CONVENTIONS.

a. 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 is restarted, any parameters previously changed are restored to their original value set at system generation time.

b. 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 the RUBOUT key or the DEL key. 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 teletype and CRT).

c. Two commas in a row means a parameter is zero.

Table 3-1. RTE Operator Commands

COMMAND FORMAT	DESCRIPTION
BR	Sets a break in named program's ID segment
OF	Turns programs off
ON	Turns programs on
RU	Starts a program immediately
UP	Declares I/O devices available
TM	Set or reset real time clock
TI	Display current year, day and time as recorded by real time clock

TABLE 3-2. CONVENTIONS IN OPERATOR COMMANDS SYNTAX

ITEM	MEANING
UPPER CASE ITALICS	These words are literals and must be specified as shown.
lower case italics	These are symbolic representations indicating the type of information 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	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."

3-6 BR

GENERAL FORM:

BR, name (where name is the program name)

EXAMPLE:

BR, FMGR

PURPOSE

This statement allows the user to interrupt a program while it is running. It also sets an attention flag in a program's ID segment.

3-7 ON

GENERAL FORM:

ON, name (where name is the program name)

EXAMPLE:

ON, EDITR

PURPOSE

This statement schedules a program for execution.

3-8 OF

GENERAL FORM:

OF, name, 1 (where name is the program being terminated)

EXAMPLE:

OF, CORED, 1

PURPOSE

Terminates a program, or removes a disc resident program which was loaded on-line but not permanently incorporated into RTE. It will also release any disc tracks used by the program.

3-9 RU

GENERAL FORM:

RU, name (where name is program name)

EXAMPLE:

RU, FMGR

PURPOSE

This statement schedules a program immediately without affecting its entry in the time list.

3-10 UP

GENERAL FORM:

UP, EQT (see note)

EXAMPLE:

UP, 6

PURPOSE

This statement declares an I/O device up (is available for use by RTE).

NOTE

EQT is the equipment table number of the device to be reenabled.

3-11 TI

GENERAL FORM:

TI

no parameters required

EXAMPLE:

TI

1980	123	10	17	28	RETURN DATA FROM OP SYST
YEAR	JULIAN DAY OF YEAR	HOUR OF DAY	MINUTE OF DAY	SECONDS	

PURPOSE

Print the current year, day and time on the CRT as recorded by the Real-Time Clock.

3-12 TM

GENERAL FORM:

TM, YEAR, DAY ,HR ,MIN ,SEC

where:

YEAR = 4 digit # for year

DAY = 3 digit # for JULIAN DAY

HR = 2 digit # for hour of the day, default is 0 (ZERO)

MIN = 2 digit # for minute of the day, default is 0 (ZERO)

SEC = 2 digit # for seconds of the day, default is 0 (ZERO)

PURPOSE

Allows user to set or reset the Real-Time Clock.

EXAMPLE:

user enter TM, 1978, 150, 14, 50, 36

date is 150th day of 1978

time is 14 hrs, 50 minutes and 36 seconds

3-13 ERROR MESSAGES.

3-14 The RTE Operating System rejects operator requests for various reasons. When a request is in error, RTE will print the messages indicated below. The operator should re-enter the request in its correct form.

MESSAGE	MEANING
OP CODE ERROR	Illegal operator request code
NO SUCH PROGRAM	The name of program used is not a main program in this system
INPUT ERROR	An illegal parameter in message string
ILLEGAL STATUS	Program is not in a appropriate state for execution

Other errors may occur when an I/O device times out because of an inoperable state. In this case, the system will print one of the following error messages:

I/O NR Lxx Eyy Szz

I/O TO Lxx Eyy Szz

where:

NR means device not ready

TO means device has timed out

Lxx = Logical unit of down device

Eyy = Equipment table number of down device

Szz = Subchannel number (if any) of down device

EXAMPLE:

I/O TO L6 E6 S0

SECTION IV

FILE MANAGER ORGANIZATION

4-1 INTRODUCTION.

4-2 The file manager is a system software package operating under the control of RTE Operating System. It operates in the background and provides the system with the following capabilities:

a. Creates named disc files through interactive operator commands or under program control

b. Creates and accesses files by name rather than by physical disc address

c. Treats peripheral non-disc devices as files for I/O control

d. Performs media conversions (i.e., transfers information from file-to-file, device-to-device, device-to-file, or file-to-device)

e. Controls I/O to and from files, or devices through program calls

f. Provides for security and file integrity or user files.

4-3 FILE ORGANIZATION.

4-4 Files are a collection of information logically organized into records. The information in files may be programs or the data used by programs. Data may be binary or in ASCII code. Programs may be in the form of ASCII source code or binary code in either relocatable or absolute form. Programs may also be in memory-image form, a form used by RTE for programs ready to be executed.

4-5 Certain terms used in discussing file organization for the file manager are defined in Table 4-1.

4-6 DISC FILE ALLOCATION.

4-7 Disc files managed by the file manager, are kept on FMGR cartridges. This cartridge is a logical entity that may correspond directly to a disc platter or may be a sub-division of the disc platter.

4-8 Each cartridge is defined by a beginning and an ending track, and is assigned a logical unit number and a cartridge reference number, either of which may be used to reference the cartridge. Files on the same cartridge must have unique names. Duplicate file names may be used as long as the duplicates are on separate cartridges so the file can be uniquely identified by its name and a cartridge identifier.

Table 4-1. File Management Terms

TERM	DESCRIPTION
disc	A rotating random access storage device on which files may be stored and from which they may be retrieved. Discs may be physically permanent or they may be removable. The system disc on logical unit 2 contains the RTE operating system.
cartridge	Sometimes used as another name for disc. It may also refer to a set of contiguous tracks on a disc platter. Cartridges contain disc files with a directory of the files stored on each cartridge. All files on the same cartridge must have unique names.
track	A subdivision of the disc
logical sector	A further subdivision of the disc; specifically, a sector is a portion of a disc track consisting of 64 words. Two sectors make up one block, the unit of information that may be physically transferred between the disc and memory.
file	A disc file is a collection of records terminated by an end-of-file mark. Non-disc devices are treated by the file manager as if the device were itself a file. The device, like a file, is a collection of records terminated by an end-of-file mark that depends on the device. Any file can have zero or more records and is designated by a name of six characters or less.
logical unit	An integer assigned to each input-output device or disc cartridge at system generation by which the cartridge or device can be referenced.
block	A subdivision of a disc file containing 128 words (two disc sectors) that is the smallest unit that can be physically transferred between disc and central memory during file access.
record	A logical collection of 16-bit words on a file or device that is terminated by an end-of-record mark. A record may have zero or more words. A record is the smallest unit that may be accessed by a call from a program.
extent	An extension to a file automatically provided by FMGR as needed. Each extent is the same length and type as the file and is identified by a positive integer called the extent number.
user buffer	An area in the calling program to hold one record at a time during file access.
ID segment	A block or words associated with each resident program. It is used by the system to keep track of the characteristics of the program.

4-9 Files are located on continuous tracks on a cartridge. User files begin in the lowest numbered track and work up. Directory entries for user files begin in the highest numbered track and work down. Removable cartridges containing FMGR files are interchangeable between drives within a system, or between drives on different systems, provided that logical track 0 refers to the same physical track on every disc unit. (Refer to Figure 4-1 for an illustration of disc organization using one cartridge on the system disc starting at the first FMGR track, and one on a peripheral disc starting at track 0).

4-10 Files may cross track boundaries. Files are subject to being moved whenever a cartridge is packed. This causes files to be relocatable within a cartridge and no absolute file addresses should be kept in any file or program.

4-11 Files always start on even sector boundaries and all accesses are multiples of 128 words addressed to even sectors.

4-12 Disc errors are passed back to the user for action. Error codes are printed on the system log device when using the file manager operator commands, or passed to the user program when calling a file management package library routine. You may report bad tracks to the system through the initialization program. Bad tracks discovered by the system result in an error return to the calling program.

4-13 DIRECTORIES.

4-14 Two directories are created and maintained by FMGR: the FMGR cartridge directory on the system disc, and the file directory on each cartridge. Only program D.RTR can write to or modify the directories.

4-15 CARTRIDGE DIRECTORY.

4-16 The cartridge directory is a master index to all FMGR cartridges. It is maintained in the first two sectors of the last track of the system disc (refer to Figure 4-1). It has an entry for all currently mounted cartridges with tracks assigned to FMGR. The directory has room to describe up to 31 cartridges using four words for each. In addition, this directory keeps the master security code for the system. With this code, the security codes for the individual files in the File Management Package can be retrieved.

4-17 FILE DIRECTORY.

4-18 A file directory, maintained on each cartridge, contains information on each file on that particular cartridge. Each directory starts in sector 0 of the last track available to the FMGR. The first 16-word entry in this directory contains label and track information for the cartridge itself. Each subsequent 16-word entry has information on a user file. The last entry is followed by a zero word. When a file is purged, the first word in the directory entry for the file is set to -1 to indicate that it is to be ignored. When the cartridge is packed, the directory entry for any purged file is cleared and the cartridge area where the file was located is over-written by non-purged files wherever possible.

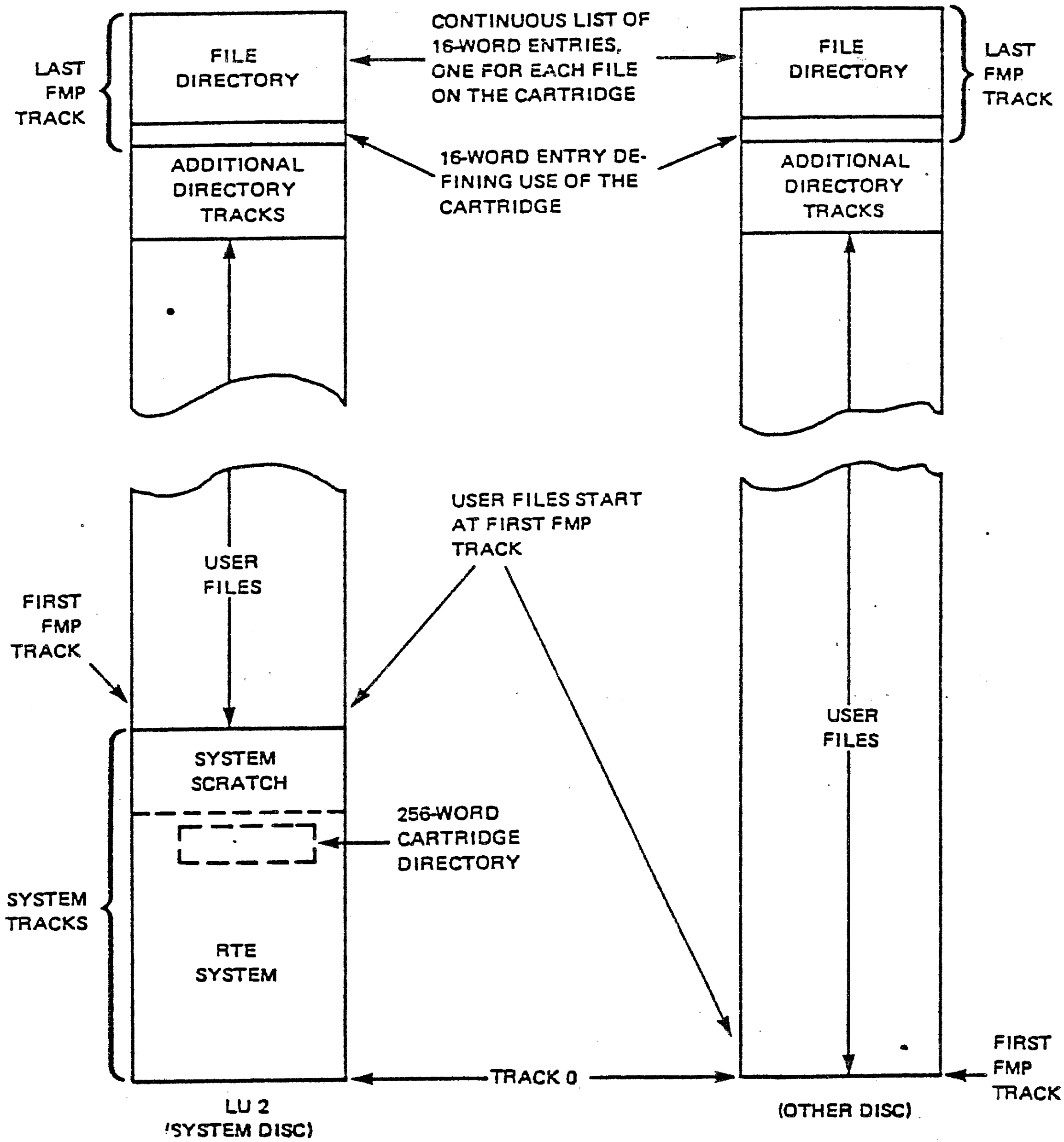


Figure 4-1. File Manager Disc Cartridge Organization

4-19 FILE TYPES.

4-20 There are three general classifications of files:

a. System files (i.e., those used by the system for housekeeping purposes.

b. ATLAS files - These files can be broken down into two classifications. They are:

   1. ATLAS source files - This type of file contains ASCII source information followed by the appropriate interpretive code.

   2. ATLAS data files.

c. User created ASCII files.

4-21 SECURITY.

4-22 File manager provides two levels of security:

a. System security

b. File security.

4-23 SYSTEM SECURITY.

4-24 During system setup, a master security code is assigned to the system. If the code is zero, no security is provided. If non-zero, the master code must be known inorder to get directory listings that include the specific file security codes and also in order to reinitialize an FMGR cartridge.

4-25 FILE SECURITY.

4-26 Each file has a security code. This code may be zero, positive, or negative. A zero code allows the file to be opened to any caller with no restrictions; in effect, this code provides zero security. A positive code restricts writing on the file but not reading; that is, a user who does not know the code may open the file for read only, but may not write on the file. A negative code restricts all access to the file; this code must be specified in order to open a file protected by it. An attempt to open a file so protected without the correct security code results in an error message.

4-27 FMGR OPERATOR COMMANDS.

4-28 Refer to Appendix D for File Manager Commands.

SECTION V

ATG POST PROCESSING PROGRAM

5-1. INTRODUCTION

5-2. This section contains the following information:

a. Concept of the ATG Post Processing Program conversion system.

b. Description of the ATG Post Processing Program conversion system.

c. Procedure for generating an ATLAS application program on the Universal Programming Station.

d. Description of the DATA ANALYZER.

5-3. CONCEPT

5-4. The ATG Post Processing Program Conversion System is a multi-function system. It consists of two programs; the CONVERTER and the ANALYZER. The CONVERTER, in itself, serves a two-fold function in that it reformats the ATG generated data files and generates an ATLAS test program. The data files are stimulus, response/mask, fault, and optionally, probe files. The reformatting (or conversion) of these files is required so that they are in a suitable form for the specific ATE. The generated ATLAS program is an ATLAS source file, which when executed by the ATLAS compiler, references these data files as well as performing the test function.

The presence of probe data on the ATG tape will be indicated by a non-zero length of LU5 in the tape table of contents (LU29). When probe data exists, WIZARD II probe data might also exist. The presence of WIZARD II probe data will be indicated by a non-zero length of LU6 in LU29. When probe data does exist, the converter will automatically process (reformat) it, create a probe data file and store all probe data and WIZARD II data (if present) on the user's disc. Automatic modification of the ATLAS test program source file to include all necessary probe-related statements is also done by the converter.

5-5. The ANALYZER is automatically invoked by the ATLAS test program whenever analysis of UUT faults is required. Operation of the ANALYZER is fully automatic and requires no user intervention.

5-6. DESCRIPTION

5-7. The ATG Post Processing Program Conversion System accepts standard output tapes generated by LOGOS/LASAR or HITS ATG's and produces an ATLAS test program with associated data files. The tape contains the following ATG Post Processing Program files: LU29, LU1, LU2, LU3, and LU4, and optionally, LU5. If the tape contains LU5 then it may optionally contain LU6 (WIZARD II probe data).

This will be 800 B.P.I. tape, with or without header, and blocked at 1600 bytes. The exact format and content of this tape is fixed and must conform with the format specified in Navy Standard Digital Automatic Test Generation Output Tape Format (NAEC-MISC-920456). The user can expect to produce all digital portions of UUT test programs by this means, but will be required to supply those portions of a test program which provide analog stimulus and measurement functions required for UUT testing.

5-8. Basically, there are three inputs to the Conversion System. They are as follows:

a. ATG saved tape

b. Pin information

c. User responses at the console

d. An optional fourth input may be required if probe data exists and more than one type of logic family is used on the UUT (for example; TTL and CMOS chips).

5-9. The Conversion System, when reading the ATG saved tape, will automatically position it to the first data file. That is, if a label exists, the tape will be positioned past the label to the first pertinent data file. Additionally, upon completion of the conversion process the tape is rewound automatically.

5-10. The pin information may be entered via several different devices. The available options for entering INPUT/OUTPUT pins are the keyboard, card reader, paper tape reader, or a type 4 disc file containing 1 DWG pin # per line (record). The option is selected at the time the RUN converter command is issued. Fixed pin capability is included to fix pins within the stimulus pattern width. To fix a pin at a "1" the user enters the pin number at the appropriate time. The Conversion System takes care of entering the pins in the ATLAS test program. To fix a pin at a "0", the operator must manually edit the ATLAS test program after conversion. The Conversion System checks each pin entry against all previously entered pins. This ensures that there are no duplications of pin numbers.

5-11. The Conversion System relieves the user from the task of creating data files and guessing the length of the files that are needed. This is accomplished automatically. The operator need only enter a maximum of five characters for the ATLAS test program name and the Conversion System creates the necessary data files with the required lengths. The naming of the data files is done by appending the letters S,R,F and P to the ATLAS program name for the stimulus, response/mask, fault and probe files respectively. Should there be a duplication of file names, the user is informed early in the conversion process so that the necessary corrective action may be taken. The resultant ATLAS test program and associated data files are generated on the top mounted cartridge. The user is responsible for ensuring that the desired cartridge is mounted on top.

5-12. ATLAS PROGRAM.

5-13. At the time of execution, the ATLAS program performs the following functions:

a. Disc data file management (e.g., FILE OPEN, FILE CLOSE)

b. Accesses the test data file and applies the test (stimuli) patterns obtained from the stimulus file.

c. Determines which patterns contain bits that do not agree with anticipated response patterns in the test data array and accesses the mask data to determine which of these represent actual failures. The failures are automatically stored in a system file called SYSERR.

d. If failures occurred, the ANALYZER is automatically invoked. Following execution of the ANALYZER, control is returned to the test program and the test process is terminated.

5-14.ATG POST PROCESSING PROGRAM OPERATING PROCEDURES.

5-15.ATG Post Processing Program operating procedures are as follows:

a. In order to use the CONVERSION SYSTEM, the user must mount his LASAR, HITS or LOGOS TAPE on the CAT-RTE Software Station tape drive. If pin data exists on either punched tape or cards, the punched tape or card deck must be inserted into the appropriate device. The CONVERSION SYSTEM reads punched tape or cards in free field format. The legend "SYS" denotes a message displayed from the system on the CRT while "KY" denotes a user's entry. UPPER CASE TYPE is used to illustrate actual information to or from the system and lowercase type for explanatory comments.

b. Replace the removable cartridge by placing the RUN/STOP switch to STOP. Wait for the DOOR UNLOCKED indicator to light and then replace the removable cartridge.

c. Replace the RUN/STOP switch to RUN and wait for the DRIVE READY indicator to light.

d. From the RTE state enter

KY*RU,CNVRT, console, pindev, psid

console - is the logical unit number of the keyboard device being used. Default is 1.

pindev - is the logical unit number for the pin list data. Default LU# is the console number, 5 is the paper tape reader, while LU 9 is the card reader. It is also possible to have the pin list read in from a type 4 disc file whose filename must be PINS and reside on either LU41 or LU42. If the disc file method is used, then the 'pindev' parameter should specify the LU # (41 or 42) where the file PINS resides.

psid - is the logical unit number of the input device that will be used to enter node voltage class modifications, if any, during probe data processing. Default is the console LU #.

SYS LASAR/LOGOS-Q CONVERTER

ENTER FILE NAME FOR ATLAS PROGRAM. DO NOT EXCEED 5 CHARACTERS.

e. KY name of ATLAS program

The CONVERSION SYSTEM generates five disc files: (A) the ATLAS TEST program file, (B) stimulus file, (C) reference-mask file, (D) fault file, and (E) the probe file which contains all information necessary for probing a UUT. The name of each data file is the ATLAS test program file name (the user supplies in step e.) concatenated with the letter S for the stimulus data file, the letter R for reference-mask file, the letter F for the fault file, and the letter P for the probe file.

f. SYS ENTER YES IF TAPE CONTAINS FAULT DATA.

KY enter YES or NO

The processing of fault data is a user option. By responding with NO to the inquiry in step g, the time required for conversion is reduced.

g. SYS ENTER DWG PIN NUMBER FOR INPUT SRA PIN XXX.

KY DWG pin number (1 _ number _ 480)

This message appears if pin list data is to be entered from the console and repeats for each INPUT and OUTPUT pin.

h. SYS ENTER YES, IF YOU WANT TO ENTER FIXED PINS.
ENTER NO, IF YOU DO NOT.

KY enter YES or NO

i. SYS ENTER FIXED PIN NUMBER.

If step h. entry was YES, the CONVERSION SYSTEM responds with a blank line for the user to enter the fixed pin number. Each fixed pin entry is set to a logic 1 and is checked for duplication against input, output, and preceding fixed pins. If a valid entry is made then step h. is repeated. The definition of a valid fixed pin entry is: a pin not already used as an input, output, or fixed pin and must reside within any 12 bit pin group of the stimulus width.

For example, if the lowest stimulus pin number is 31 and the highest stimulus pin number is 69, the range of allowable fixed pins would be between 25 on the low end and 72 on the high end. This is because pin number 31 resides in the 3rd 12-pin group (pins 25-36) and pin 69 resides in the 6th 12-pin group (pins 61-72). An attempt to fix a pin outside these limits will result in an error message being displayed on the CRT. The converter will then ask if more fixed pins are to be entered.

To set a fixed pin to a logic 0, edit the ATLAS test program and insert that pin into the stimulus connection field of the CONNECT, DIGITAL TEST statement. Steps h and i are repeated until a NO is entered.

Processing of probe data starts after all stimulus, reference-mask and fault data has been completed. The only user input that may be required is the inputting of any node names and their corresponding node type. The converter will ask the operator if there is more than one node voltage type required for the UUT by displaying (on the console) the following message.

j. <u>SYS</u> ANY NODE VOLTAGE TYPE MODIFICATIONS?

<u>KY</u> enter YES or NO

If a NO response is entered, all nodes to be probed are assumed to be type 1. Note that the converter assigns TTL logic levels (clamp voltage = 6.2 volts, Reference voltage = 1.6 volts), to the first logic type defined in the "REQUIRE, PROBE" statement in the generated ATLAS source program. See the ATLAS programming manual for further details on the "REQUIRE,PROBE" statement.

If a YES response is entered, the converter will look for the node name and its voltage type at the device specified at run time (psid). Only the console, paper tape reader, or the card reader are allowed. For example, if the card reader was used to read in the I/O pins then the card reader must also be used to input all of the node voltage type modifications. The format required is:

cols 1-8	col 9	cols 10,11
node name	blank	type

Columns 1-8 will contain the alphanumeric node name, column 9 is always blank and columns 10 and 11 will contain the new type designation (right justified) for this particular node. For example:

```
      Z14P07          2
col. 1    col. 6    col. 11

      Z21P05          3
col. 1    col. 6    col. 11
```

Only one node and its type is allowed per card (or line). As many node voltage type modifications as needed may be entered at this time. Note, however, only 10 total types are allowed. Node names must agree exactly with the original name used in the ATG model or an error will occur. If the input device was the card reader or paper tape reader the conversion process is aborted. If the input device was the console the incorrect entry is ignored, an error message is displayed describing just what is wrong with previous entry and the user is asked to re-enter the correct information.

Step (j) is repeated until all cards have been read or the user answers "NO" to the question. The converter will then finish processing all probe information, if it exists. If any node voltage types were added the user must edit the ATLAS program file "REQUIRE PROBE" statement to reflect as many new voltage types that were defined.

If WIZARD II probe data is detected on the tape in addition to the regular (WIZARD I) probe data, the next message from the system is:

k. SYS WIZARD II PROBE DATA DETECTED ON TAPE.
DATA NOW BEING PROCESSED.

The only user input that may be required is the entering of any node names and their corresponding node type, as may have been done during the processing of WIZARD I probe data. The system will display the following message:

l. SYS ARE THERE ANY WIZARD II NODE TYPE MODIFICATIONS?

From this point on, the process is exactly the same as step (j) for a YES or NO response.

The WIZARD II probe data, if it exists, is appended to the end of the previously created probe file.

The next message from the system is:

SYS CONVERSION OF

model name

COMPLETED SUCCESSFULLY.

5-16. DATA ANALYZER

5-17. No user procedures are required for operation of the ANALYZER. Each CONVERTER generated ATLAS test program will invoke the ANALYZER automatically whenever a UUT fault has been detected. When the ANALYZER is invoked, the following functions are performed:

a. Access the error file SYSERR to determine which failures occurred during testing.

b. Access the FAULT file, to obtain the set of ATG Post Processing Program predicted faults.

c. Print out an analysis of actual versus predicted failures. (Ten fault sets, in the order of best match to worst, where match is indicated by a printed weighing factor. Weighing factor of zero is an exact match).

d. Records the actual fault set number(s) in system common for possible access by the Guided Probe subsystem.

e. Transfer control back to the ATLAS program which will then terminate the test run.

5-18. Figure 5-1, presents a pictorial view of the entire automatic test program generation and execution process.

5-19. USER INSTRUCTIONS FOR PIN INSPECT/CHANGE ROUTINE

5-20. The ATG Pin Inspect/Change routine allows the user to inspect and/or change ATG generated Stimulus and Ref/Mask data files.

5-21. The ATG Pin Inspect/Change Routine is entered from the FMGR state by entering RU,PINSP. The user will then specify, in response to questions displayed on the CRT, the following information:

a. Stim or Ref/Mask File Name

b. State if File is a Stim or a Ref/Mask

c. If Ref/Mask File - Enter if pattern is Ref or Mask

d. Enter Pattern Number of interest

e. First DWG Pin # of a group to be viewed (Start Pin #)

f. # of DWG Pins to be Viewed (# of Pins).

5-22. Once the pin data is displayed on the CRT, the user has the option of changing the displayed data or displaying other DWG pin information. If a change is desired, the user enters the pin # to be changed. The CRT will respond with a new display and asks if another change is desired in the range of pins currently displayed. If not, asks if any other changes are to be made in the test pattern currently being viewed. If so, the CRT will display "ENTER 1st PIN # TO BE VIEWED, # OF PINS TO BE VIEWED, USE FORM: START PIN #, # OF PINS".

5-23. The test and pattern information may be redisplayed or any other legitimate test and pattern data may be examined. If no further display is requested, the program will terminate with the following message displayed on the CRT, "PIN INSPECT/CHANGE ROUTINE COMPLETED".

5-24. It should be noted that the user cannot be assured that the requested changes made are on the disc unless the program is terminated properly. For example, a termination would be considered improper if the user entered the FMGR state in the middle of the program. A proper termination occurs when the operator responds to all the questions displayed by the CRT in the correct manner. Having done this, the message "PIN INSPECT/CHANGE ROUTINE COMPLETED" will be displayed on the CRT.

In the event incorrect information is entered, the CRT will display an error message which will indicate which entered quantity is in error.

5-25. The following dialogue illustrates the procedure a user would follow in order to execute this program (all user inputs are underlined).

```
    *ON,FMGR
    :RU,PINSP
*******PIN INSPECT/CHANGE ROUTINE
    ENTER FILE NAME.
    CNTCDS
    IS CNTCDS A STIM OR A REF/MASK FILE?
    ENTER ST OR RM
    ST
    ST ENTERED. CNTCDS IS A STIM FILE.
    LARGEST TEST PATTERN NUMBER = 876

    ENTER PATTERN NUMBER YOUR WISH TO VIEW
    24

    ENTER 1st PIN # TO BE VIEWED, # OF PINS TO BE VIEWED.
    USE FORM:  START PIN #, # OF PINS WHEN ENTERING DATA
    11,16
    PIN # 1 - 12 000100000000
    PIN #13 - 24 100000000010
    PIN #25 - 36 000000011001

    DO YOU WISH TO CHANGE A PIN?
    ENTER Y OR N
    Y

    ENTER PIN NUMBER YOU WANT TO CHANGE
    10
    PIN # 10 IS NOT WITHIN THE BOUNDARY OF THE TEST PATTERN OR AREA OF
    INSPECTION

    DO YOU WISH TO CHANGE A PIN?
    ENTER Y OR N
    Y

    ENTER PIN NUMBER YOU WANT TO CHANGE
    12

    PIN # 1 - 12 000100000001
    PIN #13 - 24 100000000010
    PIN #25 - 36 000000011001

    DO YOU WISH TO CHANGE A PIN?
    ENTER Y OR N
    N

    DO YOU WISH TO VIEW MORE PINS IN TEST PATTERN #24?
    ENTER Y OR N
    N

    (Writes information out to disc)
```

```
ANY MORE PINS YOU WISH TO VIEW FURTHER IN FILE CNTDCS?
ENTER Y OR N
N
ANY OTHER FILES TO INSPECT OR CHANGE?
ENTER Y OR N
Y
ENTER FILE NAME
PLESYR

IS PLESYR A STIM OR A REF/MASK FILE?
ENTER ST OR RM
RM

DO YOU WANT TO LOOK AT A REF PATTERN OR A MASK PATTERN?
ENTER R OR M
R
R ENTERED.  PLESYR IS A REF/MASK FILE
LARGEST TEST PATTERN NUMBER = 2337
ENTER PATTERN NUMBER YOU WISH TO VIEW
24

ENTER 1ST PIN # TO BE VIEWED, # OF PINS TO BE VIEWED.
USE FORM:
START PIN #, # OF PINS  WHEN ENTERING DATA
125,10
PIN # 121 - 132 ------111111
PIN # 133 - 144 11----------

Note:     (Dashes represent pins of no interest)

DO YOU WISH TO CHANGE A PIN?
ENTER Y OR N
Y

ENTER PIN NUMBER YOU WANT TO CHANGE
125

PIN NUMBER 125, IS NOT A VALID PIN, TRY AGAIN

DO YOU WANT TO CHANGE A PIN?
ENTER Y OR N
Y

ENTER PIN NUMBER YOU WANT TO CHANGE
135

PIN # NOT IN BOUNDS
```

```
DO YOU WISH TO CHANGE A PIN?
N
DATA HAS BEEN WRITTEN OUT TO DISC
ARE THERE ANY MORE PINS YOU WISH TO VIEW FURTHER IN FILE
PLESYR?
ENTER Y OR N
N
ANY OTHER FILES TO INSPECT OR CHANGE?
ENTER Y OR N
N
PIN INSPECT/CHANGE ROUTINE COMPLETED
```

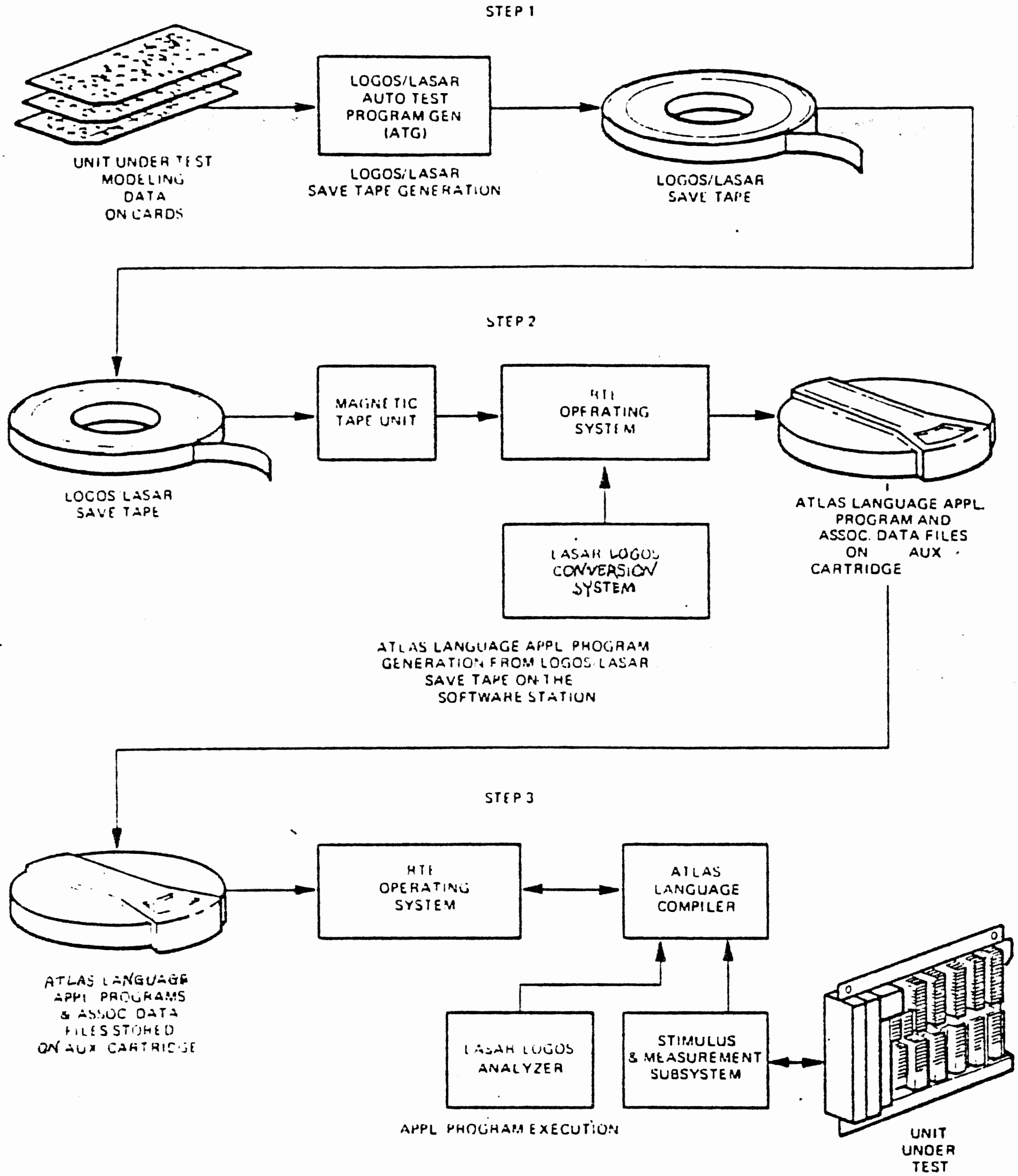


Figure 5-1. Automatic Test Program Generation Execution. Flow Diagram

SECTION VI

ATLAS DIFFERENCE DETECTOR PROGRAM

6-1 PROGRAM NAME: NCMPR.

6-2 PROGRAM DESCRIPTION.

6-3 NCMPR is a program that determines and lists the changes/additions/deletions that were made in generating a new version of an ATLAS test program from an existing program. The existing program is called the master file and the new version is the test file.

6-4 Both the master and test files must be ATLAS disc files with ATLAS source statements that can be retrieved in statement number order. The files may be:

a. Type 4 files - these are source only files generated by the 'LIST' command. All records are in statement number order.

b. Type 8 or 9 files - these files contain IC as well as source records and are generated by the 'SAVE' or 'REPLACE' commands. The segment and statement tables from the compiled program are used to retrieve the source records in statement number order.

6-5 The program uses terminators and statement numbers to determine the differences between the files. Initially, a new statement is expected in each of the input files. The processing of each record thereafter depends upon the processing that has taken place so far on each input file. It is necessary to know what type of record is expected in each file, either the start of a new statement or a continuation of a current one. This determination is based on whether or not the current statement contains a terminating character ($). The presence of the terminator in a record indicates the end of a statement. The next record in that file is then expected to be the start of a new statement and should contain a statement number. If the terminating character is missing, the next record is a continuation of the current statement. Continuations do not contain statement numbers.

6-6 When statement numbers are expected in both files, these numbers are compared. This comparison determines changes/additions/deletions to the original file. When the statement numbers are equal, a record comparison is made. If the records have different lengths or are unequal, a change is indicated and the records are printed. If the master file statement number is greater than the test file statement number (statement numbers are in sequential order), an addition is indicated and the test file record and remainder of statement are printed. When the master file statement number is less than the test file statement number, a deletion is indicated and the master file record and remainder of statement are printed.

6-7 PROGRAM EXECUTION.

6-8 To start program execution, the user inputs RU, NCMPR on the terminal. No other information or parameters are needed.

6-9 INPUTS.

6-10 When the program is run, a message from the terminal requests the user to enter a master filename: 'ENTER MASTER FILENAME &:SECURITY CODE &:CARTRIDGE**.'

6-11 After the master filename is entered on the terminal, another message requests the user to enter the test filename: 'ENTER TEST FILENAME &:SECURITY CODE &:CARTRIDGE**.' As mentioned earlier, both files must be type 4, 8 or 9 ATLAS disc files, although both need not be the same.

6-12 OUTPUTS.

6-13 The program outputs the differences between the two files to the line printer. The following format is used:

a. Records do not compare (record length/contents differ)

   o print 'CHANGE:'

   o print master record

b. Addition (record in test file but not in master)

   o print 'ADDITION:'

   o print test record and remaining records in statement

c. Deletion (record in masterfile but not in test)

   o print 'DELETION:'

   o print master record and remaining records in statement.

6-14 Error messages are output to the terminal or to the line printer. The nature of the error determines this, as well as whether or not the program will continue execution. The program processes errors as follows:

6-15 ERROR PROCESSING.

6-16 INVALID FILE NAME (invalid character in name, too long)

a. display error message on terminal: 'INVALID MASTER/TEST FILENAME;

b. repeat request for master/test filename: 'ENTER MASTER/TEST FILENAME &:SECURITY CODE &:CARTRIDGE**'

6-17 UNSUCCESSFUL FILE OPEN.

a. display error message on terminal: 'MASTER/TEST FILE UNSUCCESSFULLY OPENED FMGR-OXY' (where -OXY is the error code encountered during the open)

b. repeat request for master/test filename: 'ENTER MASTER/TEST FILENAME &:SECURITY CODE &:CARTRIDGE** '

6-18 READ RECORD ERROR (any record in ATLAS disc file)

a. display error message on line printer: 'READ MASTER/TEST RECORD ERROR: FMGR-OXY' (where -OXY is error code encountered during the read)

b. close any open files and terminate pgm.

6-19 NO STMT # WHEN EXPECTED (or no test #)

a. display error message and statement where error occurred: 'NO STATEMENT NUMBER ENTERED FOR MASTER/TEST REC (master/test statement)'

b. process next record.

6-20 INVALID STATEMENT NUMBER (invalid characters)

a. display error message and statement where error occurred on line printer: 'INVALID STATEMENT NUMBER FOR MASTER/TEST REC (master/test statement)'

b. process next record.

6-21 INVALID SECURITY CODE (encountered during file open).

a. display error message on terminal: 'INVALID SECURITY CODE'

b. repeat request for master/test filename: 'ENTER MASTER/TEST FILENAME &:SECURITY CODE &:CARTRIDGE**'

6-22 INVALID CARTRIDGE NUMBER (LU where file found).

a. display error message on terminal: 'INVALID LOGICAL UNIT'

b. repeat request for master/test filename: 'ENTER MASTER/TEST FILENAME &:SECURITY CODE &:CARTRIDGE**'

6-23 INVALID FILE TYPE (only type 4, 8 or 9 files allowed).

a. display error message on terminal: 'INVALID FILE TYPE-NOT TYPE 4, 8, 9'

b. repeat request for master/test filename: 'ENTER MASTER/TEST FILENAME &:SECURITY CODE &:CARTRIDGE**'

6-24 SAMPLE OUTPUT (Table 6-1)

6-25 NCMPR results using a type 4 master file, M89, and a type 8 test file, T8.

Table 6-1. Sample Output

```
MASTER FILE    : M89
TEST FILE : T8
CHANGE:
     000012 DECLARE,COMMON,INTEGER,STORE,'S','C','D','E' $
     000012 DECLARE,COMMON,INTEGER,STORE,'C','C','E','E' $
ADDITION:
     000014 DECLARE,COMMON,DECIMAL,
               STORE,'W' S
CHANGE:
     000015 DECLARE,COMMON,INTEGER,LIST,'TEST'(20),'G'(8),'R'(8),'F'(8)$
     000015 DECLARE,COMMON.
ADDITION:
                                   MSGCHAR,LIST,'PARAM'(8),8 CHAR $
DELETION:
     000014 DECLARE,COMMON,MSGCHAR,LIST,'PARAM'(8),8 CHAR $
CHANGE:
     000020 DECLARE,INTEGER,STORE,'P','K','X','J','L1','L2' $
     000020 DECLARE,INTEGER,STORE,'P','K','X','J','L1','L2','L3' $
CHANGE:
     000025 DECLARE,DECIMAL,STORE,'W' $
     000025 DECLARE,DECIMAL,STORE,'WW' $
DELETION:
     0000200   *
               * THIS PROCEDURE INPUTS A NUMBER FROM THE KEYBOARD
               *  AND VERIFIES THAT IT IS AN INTEGER IN THE RANGE
               *   'UL'  ='INPUT'  ='LL'
               * PROCEDURE RETURNS WITH 'GO'/'NOGO' FLAGS SET
                  ACCORDINGLY.
               $
CHANGE:
         05 DEFINE, 'NUMIN', PROCEDURE ('UL','LL') RESULT ('INPUT') $
     000205 DEFINE, 'NUMIN', PROCEDURE ('UL','LL') RESULT ('INPUT') $
CHANGE:
         15 DECLARE, DECIMAL, STORE, 'INPUT' $
         15 DECLARE, INTEGER, STORE, 'INPUT' $
ADDITION:
     000235 COMPARE, 'TESTIT', EQ 'UL' $
DELETION:
     000800 DEFINE, 'ALLOK', MESSAGE,

      ADJUSTMENT OF DAC #(1) HAS BEEN SUCCESSFULLY COMPLETED!!$
DELETION:
     000910 CALCULATE, '3'=0 $
DELETION:
     000920 CALCULATE, '0'=0 $
```

APPENDIX A

RTE INTERACTIVE EDITOR

A-1 INTRODUCTION

There are two editors provided, EDITR and EDIT/1000. This section will detail EDITR. Refer to the "EDIT/1000 User's Guide" Part No. 92074-90001 for a description of EDIT/1000.

The RTE Interactive Editor (EDITR) is commonly used for:

- o Creation of new programs or data files in ASCII code.
- o Modification of new or existing ASCII files.
- o Merging several ASCII files to each other to form a single file.

EDITR operates in either interactive or in batch mode. When used interactively, EDITR accepts operator commands from a keyboard device. When used with batch jobs, EDITR commands are included as part of the job command file.

A-2 EDITR WORK AREAS

EDITR references a source file and an output file and uses two temporary work areas on disc. The source file is read into a work area on disc called the source work area. Edited text is passed to another work area, also on disc, called the destination work area. An editing pass is completed when EDITR has read and passed all of the text lines in the source work area. Before another editing pass can be made, the source work area is replaced by the old destination work area, and a new destination work area is designated.

If the user backs up to edit a line preceding the current line, the editing pass is completed. The remaining information in the source work area is passed to the destination work area and the "destination work area" now becomes the "source work area".

When editing is complete and the EDITR is terminated with one of the terminate commands, the remaining data in the source work area is passed to the destination work area, and the destination work area is written to an output file. The relationship between these files and work areas is shown in Figure A-1.

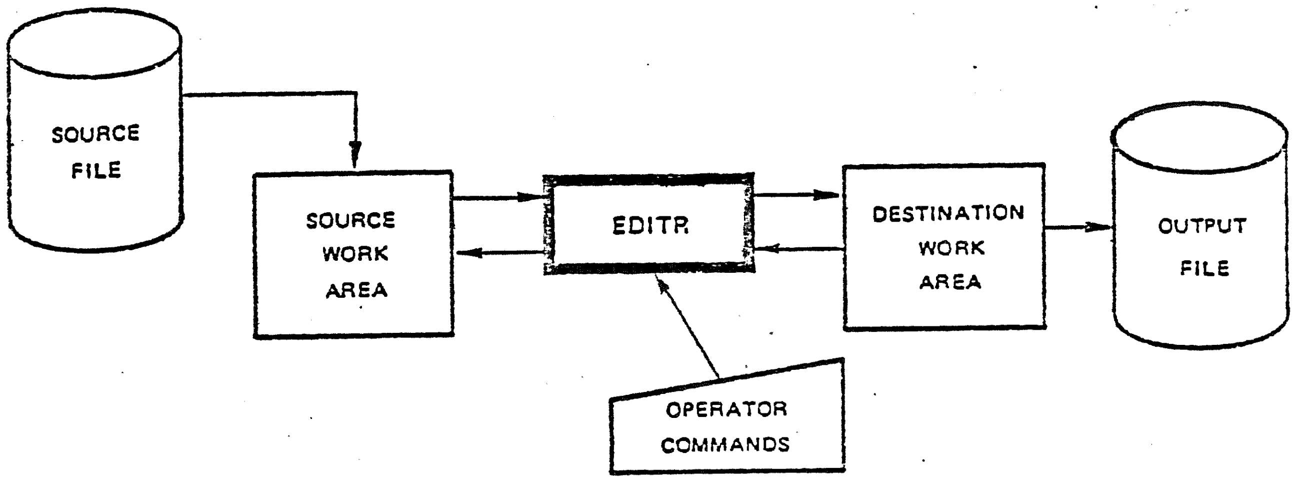


Figure A-1. File/Work Area Relationship

A-3 PENDING LINE

When EDITR is run, and the source file the user names at the beginning of the editing session is read into the source work area, the first line is displayed. This displayed line is the current line available for editing and is called the "pending line". This line remains the pending line until the user requests a new pending line with an EDITR command. In this way the user can continue to re-edit the same line until he is satisfied.

When the user requests a particular line of text, EDITR searches through the source work area until the requested line is encountered. That line then becomes the new "pending line" and is displayed on the user's terminal. EDITR maintains a pending line pointer into the source work area.

When the new pending line is displayed, the old pending line, and all other lines passed over by the pointer in the search for the new pending line, are usually written to the destination work area (see the Jump command for an exception), and are no longer accessible until the "destination work area" becomes the "source work area". In other words, the user cannot go from line 0028 back to line 0024 without the source work area being replaced by the destination work area. Generally the user does not have to be concerned with this exchange because it is done automatically. However, the user does need to remember that once the "source work area" is replaced, the previous "source work area" is totally gone along with any lines deleted or changed in the last pass of EDITR. The user should remember that if he made any insertions or deletions, these lines are no longer accessible by their original line numbers. For example, if line 3 is deleted in one editing pass, the original line 4 will be the new line 3. EDITR permits lines of text to be accessed without using line numbers.

A-4 KEYBOARD CONVENTIONS

Keyboard commands are used to direct EDITR to do replacements, insertions, deletions, searches and exchanges of text. These functions can be performed on characters within a line; they can be used to manipulate one entire line; or they can be used on groups of lines.

To use EDITR efficiently, the user must know the command syntax and keyboard conventions it expects. The command syntax used in this manual is summarized in Table A-1.

Conventions Used in Examples:

( )	Space entered at the keyboard.
CNTL/X	Indicates a nonprinting control character (in this case, X) entered at the keyboard.

Table A-1. Command Syntax

CONVENTION	MEANING	COMMAND	EXAMPLE	COMMENTS
UPPER CASE LETTERS	Literals that must be specified as shown	F	F	No Parameters
lower case letters	Variables to be replaced by values as defined in text	Wa,b	W7,9	Constants 7 and 9 replace variables a and b
CNTL/	Nonprinting control characters entered by pressing the CNTL key and another key simultaneously	CNTL/key	CNTL/C	
[ ]	Bracketed parameters are optional; if omitted default values are supplied			filename[:sc[:crn]] example: MYFILE:AA:2

A-5 DISPLAY FORMATS

The pending line displayed and listings output by EDITR are always preceded by two blanks. This convention allows room for the EDITR prompt and a single character command, and results in the new text being automatically aligned with that displayed by EDITR. Error messages, in contrast, always begin in column 1. This is so that the difference between an EDITR message and a line of text can be easily seen. For example:

```
  EOF     EDITR text
EOF       EDITR message
```

A-6 CALLING EDITR

The user can call EDITR with either the File Manager or System RU commands. The File Manager (FMGR) RU command is described in Appendix D and the System RU command is described in Section III.

The two optional parameters passed to EDITR when it is scheduled are:

lu - the logical unit number of the device to be used for command input. The default for this parameter is the user's terminal.

line length - the maximum output line length, in characters; where the default and maximum size is 150 characters. Any line longer than 150 characters, or the specified length, will be truncated.

For example, to schedule EDITR from the File Manager with commands input from the user's terminal and lines limited to 72 characters, the command would be:

```
:RU,EDITR,,72
```

LINE LENGTH

The user can specify a line length which is compatible with the device to which his output file is going to. Otherwise, long lines may be truncated. Table A-2 shows the output commonly used with EDITR and the number of characters per line each supports. The user should remember that because of the two space convention used in displays, printed lines may be 74 characters.

Table A-2. Maximum Line Lengths for Common Devices

DEVICE	LINE LENGTHS	COMMENTS
Punched Cards	80 Characters	
CRT	72 - 80 Characters	
Magnetic Tape	150 Characters	Default maximum length supported by EDITR
TTY Device	72 Characters	Longer lines will encounter printing problems
Paper Tape	150 Characters	Default maximum length supported by EDITR
Disc	150 Characters	Default maximum length supported by EDITR
Line Printer	80 - 132 Characters	Number of characters varies with printer model. Consult appropriate manual for your printer.

A-7 EDITR PROMPT CHARACTER

When EDITR is used in the interactive mode, it prompts for input with a slash (/). The X command (refer to X (Change EDITR Prompt Character) section) can be used to change this prompt character to any other character. Thereafter, the specified character is output as the EDITR prompt.

A-8 FILE DEFINITION

The NAMR used by the EDITR is a simplified version of the File Manager NAMR (refer to Section V for description). The EDITR version of NAMR is defined to be a file name followed by two subparameters. The subparameters may be omitted from the end of the list. If an embedded subparameter is omitted, its position must be indicated by the colon (:). The NAMR format is:

```
filename [:security code[:cartridge reference number]]
```

See NAMR description in Section V for explanation of parameters.

A-9 EDIT EXISTING FILE

When EDITR begins execution, it requests information about the file to be edited and prompts the user:

```
SOURCE FILE?
/
```

At this point, there are four legal responses the user can make, each of which must be followed by a carriage return:

1. 0 (zero)
2. : (colon)
3. filename
4. ( ) (blank)

If "0" or a blank is specified, the user will automatically begin working with an empty file. The EDITR commands can then be used to enter and edit lines of text. When creating a file, this is the recommended response. See the CREATE FILE WITH EDITR section for an illustrative example.

If ":" is specified, the EDITR is immediately aborted and control returns to the program from which the EDITR was scheduled.

If a file name is specified, the contents of the file will be copied into the EDITR's source work area. The number of characters per line in the named file cannot be greater than the current maximum line length.

The file may be a type 0 file to read source information directly from a peripheral device. The file name may be specified with or without a security code and a cartridge reference number. For example, to specify a file without a security code, but with a cartridge reference number:

```
/FILE1::27
```

A-10 CREATE FILE WITH EDITR

To create a file with the EDITR, enter a 0 (zero) in response to the EDITR prompt "SOURCE FILE?". EOF is printed and the following lines can be entered as illustrated:

```
:RU,EDITR
SOURCE FILE?
/0 _ _ _ _ _Enter 0 to put empty file into EDITR's source work
            area.

EOF
/ NEW FILE_ __Enter space to add line of text following pending line,
            then enter line of text to be added.

/ LINE TWO
/ LINE THREE
/ECFILE1__ _ __End EDITR and create FMP file named FILE1.
```

The above command sequence will enter the three lines of text shown into the newly created file named FILE1.

A-11 EDITR TERMINATION

The EDITR terminate commands (EC and ER) usually assign the final version of text in the destination work area to a File Manager type 4 file. The destination work area can also be output directly to a device through a type 0 file.

A-12 EDITR COMMANDS

All of the EDITR commands are summarized in Table A-3. The commands are broken down into functional groups to facilitate referencing.

The CONTROL COMMANDS provide additional EDITR features beyond the normal text manipulation functions.

The DISPLAY COMMANDS cause the contents of the source work area or information about the contents to be displayed. The display normally occurs on the user's terminal, but some commands allow displays on the line printer and other logical units.

The LINE EDIT COMMANDS allow the user to manipulate one entire line of text at a time. In addition to these EDITR commands, certain keys on the keyboard such as DEL can also be used.

The DEL key is used to delete a line if pressed before the RETURN key enters the line into the destination work area. It prints a back slash (\\), and causes a line feed and a carriage return to the start of the next line (prompt is not repeated) where the correct line can be entered.

The CHARACTER EDIT COMMANDS provide a means to change the contents and modify the structure of the current line of text. Four commands allow the replacement, insertion, and deletion of characters. Each of these commands uses nonprinting control characters so that alignment is maintained in the text. Each must be used with an initial "P", "C", or "O" command.

The current delimiter is used as a "place holder" to preserve existing text in the pending line. The P, C and O commands are not themselves character edit functions. They are used to determine the line disposition during the edit: P leaves the edited line as the pending line; C advances the pending line to the next line after the edit; O sends the pending line to the destination work area and then edits a copy of the line and leaves it pending.

Special considerations apply to these commands in a multipoint environment. Refer to the EDITR IN A MULTIPOINT ENVIRONMENT section for a complete explanation on how to use the EDITR in that environment.

The PATTERN EDITS involve multiple lines of text from the source work area. There are two kinds of pattern edits: searches and exchanges.

A search matches a character string called a "find field" with a corresponding string in the source work area. There are four search commands (B, D, F, and J) which differ by where they begin the search and the length of the data block which they search.

Exchanges consist of two sets of character strings entered at the keyboard. Whenever the first string is encountered in the source work area, it is replaced by the second string. The commands used to perform exchanges are: G, Y, X, Z, V, and U.

The TERMINATE COMMANDS are used to end EDITR operations. With the exception of the Abort command, they perform any enabled exchanges, cause data remaining in the source work area to go to the destination work area, and cause the edited destination work area to be written to the output file.

All of the normal EDITR termination commands begin with E. Once E is entered EDITR begins its termination process. Only the second letter of a normal terminate command or DEL can be entered at this time.

Output to a device of a type 0 file can only be done using the ER terminator since EDITR will not create a type 0 file using the EC.

Table A-3. EDITR Command Summary

FUNCTION	COMMAND	DESCRIPTION	PAGE NO.
CONTROL	X	Changed EDITR Prompt Character	A-10
	CNTL/G	Bell Control	A-11
	T	Set Tab Stops	A-12
	W	Set Window	A-13
	#	Sequence Numbers	A-13
	=	Set Line Length	A-16
	K	Kill Trailing Blanks	A-17
	Mnamr	Merge Source File Following Pending Line	A-19
DISPLAY	Ln	Display a Number of Lines	A-21
	n	Display Specified Line	A-23
	/ or +	Space Down a Number of Lines and Display	A-24
	N	Display Pending Line Number	A-26
	ND	Display the Number of Lines in the Destination Work Area	A-27
	H	Display the Number of Characteristics In the Pending Line	A-28
	HL	Display Header	A-30
	^	Back up in Destination Work Area	A-31
	S	Display Approximate Number or Words in the Destination Work Area	A-32
	P	Display Pending Line	A-33
LINE EDITS	P	Edit and Display Pending Line	A-33
	C	Edit Pending Line and Advance Line	A-34
	O	Duplicate Pending Line and Edit	A-35
	R	Replace Pending Line with Text	A-37
	I	Insert Text before Pending Line	A-38
	( )	Insert Text after Pending Line	A-39
	-	Delete a Number of Lines	A-40
CHARACTER EDITS	CNTL/R	Replace Characters	A-41
	CNTL/I	Insert Characters	A-42
	CNTL/S	Insert Characters	A-42
	CNTL/C	Cancel Characters	A-43
	CNTL/T	Truncate Remainder of Pending Line	A-44

Table A-3 (Continued)

FUNCTION	COMMAND	DESCRIPTION	PAGE NO.
PATTERN EDITS	Search Commands		
	;	Find Tab Field	A-45
	esc	Find Field of Indefinite Length	A-45
	/	Find Field within Window	A-45
	CNTL@	Find a Zero Length Line within this Field	A-45
	B	Find a Line with Find Field SOF to EOF	A-45
	F	Find a Line with Find Field Pending Line to EOF	A-47
	D	Delete Lines to Find Field or EOF	A-50
	J	Jump to Find Field and Make It New Pending Line	A-54
	Exchange Commands		
	G	Character Replace on Pending Line	A-56
	Y	Exchange on Pending Line, Display Next Occurrence of Pattern	A-57
	X	Enable Exchange Pattern All Lines (list)	A-60
	Z	Enable Exchange Pattern for Next Edit (no list)	A-62
	V	Unconditonal Exchange (list)	A-64
	U	Unconditonal Exchange (no list)	A-67
TERMINATE	A	Abort EDITR	A-69
	ECnamr	Create File Manager File	A-70
	ER	Replace Old File with New File Do Not Change Name	A-71
	ERnamr	Replace Old File with New File Change Name	A-71

A-13 X (CHANGE EDITR PROMPT CHARACTER)

Changes the EDITR prompt character (default is slash) to a user defined prompt character.

Xx

---

x
New prompt character (default is slash).

EXAMPLE:

/X$	Changes the prompt from a slash (/) to a dollar sign ($).
$	New EDITR prompt.

COMMENTS:

If the prompt character is changed, then the new prompt is required as the delimiter between the fields of an exchange command (i.e., for X, Y etc. commands) and for each character to be preserved when doing a character edit (i.e., in P, C, and O commands).

The space down command (/) is not affected by changing the prompt character.

The default for the EDITR prompt is always a slash at the start of an edit session regardless of what it might have been changed to in a prior edit session.

A-14 CNTL/G (BELL CONTROL)

Turns terminal bell on or off. When EDITR is scheduled, on a terminal with a bell, the bell is rung automatically every time a prompt is displayed.

- -

CNTL/G

- -

A control G is input by striking the "G" key while depressing the "CNTL" key on the terminal. It is a nonprinting character.

- -

COMMENTS:

The bell control only pertains to the period within an edit session. The bell is always ON when the edit session starts, even if it had been turned OFF in a prior session and not turned back on.

A-15 T (SET TAB STOPS)

Changes the EDITR tab character (;) and the default tab stop (7th and 21st columns) to user defined values.

- -

Tx ——— Change tab control character, leave stops.

Tts1,s2,...,s10 —— Change tab stops, leave control character.

Txs1,s2,...,s10 —— Change tab stops and control character.

- -

x
New tab control character (replaces the original semicolon or current tab control character).

t
Current tab control character

s1,s2,...,s10
New column numbers of the tab stops (replacing default values of 7 and 21). The maximum number of tab stops that can be defined at one time is 10.

- -

EXAMPLES:

1. /T%4,9 ——— Changes the tab character to a percent sign (%) and changes the tab stops to columns 4 and 9.

   / %A%B —— Command to add a line with an "A" in column 4 and a "B" in column 9.

   /P ——— Command to display pending line.

   A B Displayed pending line showing "A" in column 4 and "B" in column 9.

2. /T;11,22 —— Changes tab stops to columns 11 and 22 without changing tab control character from semicolon (;).

COMMENTS:

Tabs used beyond the highest defined stop are replaced with blanks. For instance, using the above example:

/ %A%B%C%D Command to add line with letters A, B, C, and D at tab stop locations.

/P Command to display pending line.

A B C D Displayed pending line showing "A" in column 4 and "B" in column 9 (the defined tab stops). Since no tab stops have been defined beyond column 9, "C" and "D" follow with one blank space preceding each of them.

When EDITR is called, the tab character and tab stops are defaulted to a semicolon (;) and columns 7 and 21 even though they may have been redefined in a prior edit session and not changed back.

In a Multipoint environment, special considerations are necessary for setting tab stops. Refer to the EDITR IN A MULTIPOINT ENVIRONMENT section for details.

A-16 W (SET WINDOW)

Allows user to limit the area of each record which is searched for a character string to be found, and/or exchanged.

- -

Wa,b

- -

a
Initial column number of window (default is 1).

b
Final column number of window (default is 150).

- -

EXAMPLE:

/W7,9 The window is set to include only columns 7, 8 and 9.

COMMENTS:

The set window command is especially useful when used in conjunction with the search commands (F, B, D, and J) and the exchange commands (G, Y, X, Z, V, and U). See the X command (for exchanging string patterns) for an example of using the set window command.

When EDITR is called, the display field or "window" consists of columns 1 through 150 even if the set window command had been invoked to define another window in a prior edit session.

A-17 # (SEQUENCE NUMBERS)

Allows user to place sequence numbers (in columns 76 through 80) on all lines in a file. Also, a three column identifier (in columns 73 through 75) can be specified by the command.

- -

[xxx [n1[,n2]]]

- -

xxx
Three character identifier. It occupies columns 73-75 and must be included when specifying n1 and n2 (it may contain blanks).

(Sequence Numbers)

n1
Starting sequence number. If omitted, numbers start with 00000.

n2
Incrementing value for sequence numbers. n1 is incremented by n2 for each line.
If omitted, n1 is incremented by 10.

- -

EXAMPLES:

Given a file containing the following three lines:

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE
THIS IS LINE TWO OF SEQUENCING EXAMPLE
THIS IS LINE THREE OF SEQUENCING EXAMPLE
```

any one of the following four commands can be given (in all the examples it is assumed that line 0001 is the pending line). The resulting edited file is also shown.

1. /# ----- Sequence lines with no three character identifier and default values for n1 and n2.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                        00000
THIS IS LINE TWO OF SEQUENCING EXAMPLE                        00010
THIS IS LINE THREE OF SEQUENCING EXAMPLE                      00020
```

2. /#ABC ----- Sequence lines with ABC as the three character identifier with default values for n1 and n2.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                     ABC00000
THIS IS LINE TWO OF SEQUENCING EXAMPLE                     ABC00010
THIS IS LINE THREE OF SEQUENCING EXAMPLE                   ABC00020
```

3. /#ABC,1 ----- Sequence lines with ABC as the three character identifier, the first line starting with 00001 and n2 defaulted to 10.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                        ABC00001
THIS IS LINE TWO OF SEQUENCING EXAMPLE                        ABC00011
THIS IS LINE THREE OF SEQUENCING EXAMPLE                      ABC00021
```

4. /#ABC,1,1 ----- Sequence lines with ABC as the three character identifier, first line starting with 00001 and subsequent lines incremented by 1.

```
THIS IS LINE ONE OF SEQUENCING EXAMPLE                        ABC00001
THIS IS LINE TWO OF SEQUENCING EXAMPLE                        ABC00002
THIS IS LINE THREE OF SEQUENCING EXAMPLE                      ABC00003
```

COMMENTS:

When listing a file on the terminal or the line printer, sequence numbers may be lost due to truncation. To delete the blanks in columns 60-70 so that the entire sequence number field can be output, these commands are entered at the terminal:

```
/W60, 70        -------Set window to delete blanks.
/Z          /   -------Exchange 10 blank spaces for no blank
                       spaces (see Z exchange command).
/3              -------Number of lines to be changed (required
                       for Z command).
```

A-18 = (SET LINE LENGTH)

Allows user to reset the line length to another value and to truncate any characters in the line beyond the new limit.

- -

```
=n
```

- -

n
Number of columns in line length.

- -

EXAMPLE:

Given a file containing the following two lines:

```
THIS IS LINE ONE OF THE LINE LENGTH EXAMPLE
THIS IS LINE TWO OF THE LINE LENGTH EXAMPLE
```

The set line length command can be issued to truncate the lines to eliminate everything beyond column 8.

```
/1     ----Makes line one pending line.
/=8    ----Changes line length to 8 columns.
```

The resulting file will look like:

```
THIS IS
THIS IS
```

If a third line was input at this point,

```
/ THIS IS LINE THREE OF THE LINE LENGTH EXAMPLE
```

The line would be truncated to include only the first 8 columns

```
/P            ---Display pending line.
  THIS IS     ---Truncated third line.
```

COMMENTS:

The line length initially defaults to 150 characters when the EDITR is turned on unless a different line length is passed through the parameter string in the RU, EDITR command (see the CALLING EDITR section).

The line length command is useful for such things as eliminating line sequence numbers.

A-19 K (KILL TRAILING BLANKS)

Deletes trailing blanks from all lines in the work area.

K

No parameters required.

EXAMPLE:

Given a file with the following three lines of text:

```
LINE ONE OF KILL EXAMPLE                                             00000
LINE TWO OF KILL EXAMPLE                                             00010
LINE THREE OF KILL EXAMPLE                                           00020
```

If the sequence numbers are eliminated via the set line length command,

```
/=72      ---Columns 73 through 80 will be truncated.
```

the blanks up to column 72 will remain thus requiring each record in the file to be 72 characters.

```
/H        ---Display number of characters in pending line.
  72      ---Number of characters in pending line.
```

To eliminate these unnecessary blanks from the file, the kill trailing blanks command can be used.

```
/1        ---Make line one the pending line.
/K        ---Kill trailing blanks in the file.
```

The resulting lines of text will then be stored more efficiently by making each record (or line of text) only as long as necessary.

```
/1        ---Make line one the pending line.
/P        ---Display pending line.

LINE ONE OF KILL EXAMPLE

/H        ---Display number of characters in pending line.
  24      ---Number of characters in pending line (always even).
```

COMMENTS:

When a line of text is input through the EDITR, only the characters input prior to the carriage return are stored in a record. If the line

    / LINE ONE OF KILL EXAMPLE

had been input and immediately followed by a carriage return, there would only be 24 characters on that line. The kill trailing blanks command would not be necessary in this case.

The kill trailing blanks command always causes the source work area to be replaced by the destination work area, and always ends with an EOF message.

A-20 M (MERGE SOURCE FILE)

Allows user to merge a specified file after the pending line.

- -

Mnamr

- -

namr
file name [:security code [:cartridge reference number]]

(Refer to Appendix D for the description of NAMR parameter.)

- -

EXAMPLE:

This example will merge two files, a source file called FILE1 and a file to be merged called FILE2.

Following is a listing of the source file, FILE1:

```
  :LI, FILE1    -----FMGR command to list FILE1.

FILE1  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE OF SOURCE FILE.
0002 LINE TWO OF SOURCE FILE.
```

Following is a listing of the file to be merged, FILE2:

```
  :LI, FILE2    -----FMGR command to list FILE2.

FILE2  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE OF FILE TO BE MERGED.
0002 LINE TWO OF FILE TO BE MERGED.
```

To merge FILE2 at the end of FILE1, the following commands should be input at the terminal.

```
  :RU, EDITR                    ---FMGR command to run EDITR.
  SOURCE FILE?                  ---EDITR response requesting file to be
                                   edited.
  /FILE1                        ---Name of file to be edited.
    LINE ONE OF SOURCE FILE.    ---First line of FILE1.
  //                            ---EDITR command to advance pending line by
                                   one line.
```

```
  LINE TWO OF SOURCE FILE.  ---New pending line.
 /MFILE2                    ---EDITR command to merge FILE2 after pending
                               line.
 EOF
 /ER                        ---EDITR command to end edit session,
                                 replacing original version of FILE1
                                   with edited version.
 END OF EDIT
```

The edited version of FILE1 can now be listed to show the effect of the merge command.

```
 :LI, FILE1                 ---FMGR command to list FILE2.

FILE1  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE OF SOURCE FILE.
0002 LINE TWO OF SOURCE FILE.
0003 LINE ONE OF FILE TO BE MERGED.
0004 LINE TWO OF FILE TO BE MERGED.
```

COMMENTS:

The file to be merged can be merged in after any desired point in the source file. The merge takes place after the pending line and before the next line. The next line becomes the new pending line.

A-21 L (DISPLAY A NUMBER OF LINES)

Displays a specified number of lines, starting with the pending line.

- -

```
Ln [,lu]
```

- -

n
Number of lines to be printed (starting with pending line).

lu
Logical unit number of list device; default is user's terminal.

- -

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

While on line one as the pending line, the first two lines can be displayed by the following command:

```
/L2            ---EDITR command to display two lines of text.
  LINE ONE.    ---First line of text displayed.
  LINE TWO.    ---Second line of text displayed.
```

COMMENTS:

When a file is listed on the line printer using the File Manager LI command (see Section V for an explanation of this command), line numbers will precede each line of text. Using FILEA above as an example:

```
:LL,6          ---Change list device to line printer (LU 6).
:LI, FILEA     ---FMGR command to list file.
```

The display on the line printer will look like:

```
FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001 LINE ONE.
0002 LINE TWO.
0003 LINE THREE.
```

If it desired that the file heading and line numbers not be listed, the EDITR "L" command can be used to list the file. In addition, the optional parameter, lu, can be specified to list the file on some devices other than the default list device which is the user's terminal.

If only the pending line is desired to be displayed on the user's terminal, the EDITR P command may also be used. Using the example above to demonstrate:

```
/1            ---Make line one pending line and display it.
  LINE ONE.
/P            ---Display pending line.
  LINE ONE.
```

A-22 n (DISPLAY A SPECIFIED LINE)

Displays the requested line and makes it the pending line.

```
n
```

---

n
The line number of the line to be displayed and made pending line.

EXAMPLE:

Given FILEA containing the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

Assuming the pending line is the first line in the file and the third line is desired to be displayed and made the pending line, the following command can be issued:

```
/3                  ---EDITR command to display third line of file
                       and make it pending line.
  LINE THREE.       ---Displayed third line of file.
/P                  ---Display pending line.
  LINE THREE.       ---Displayed pending line.
```

COMMENTS:

If the line number requested is less than or equal to the current pending line number, the destination work area replaces the source work area, changing text line numbers, if any insertions or deletions were made to the text.

A-23 / or + (SPACE DOWN A NUMBER OF LINES)

Advances the pending line the specified number and displays the new pending line.

```
+[n[,lu]]

   or

/[n[,lu]]
```

---

n
The number of lines to be skipped (default is one).

lu
Logical unit number of device on which the display occurs (default is the user's terminal). EDITR only recognizes this parameter when command is used in conjunction with an exchange command (G,Y,X,Z,V, and U).

EXAMPLE:

Given FILEA containing the following three lines,

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode and line one is the pending line, you can make line three the pending line and display it by using the space down a number of lines command.

```
/P          --------------Display pending line.
  LINE ONE. --------------Pending line displayed.
/+2         --------------Skip two lines and display pending line.
  LINE THREE.-------------Pending line displayed.
```

The same thing can be done by using the slash(/) instead of the plus (+) sign.

```
/P          --------------Display pending line.
  LINE ONE. --------------Pending line displayed.
//2         --------------Skip two lines and display pending line.
  LINE THREE.-------------Pending line displayed.
```

SPACE DOWN A NUMBER OF LINES (Continued)

COMMENTS:

The optional lu parameter can be used in conjunction with an exchange command. The following commands would be an example of this:

```
/1                    --------------Makes line one pending line.
  LINE ONE.           --------------Pending line displayed.
/XLINE/LINE NUMBER    --------------Exchange each occurence of the word
                                    LINE with LINE NUMBER.
/+3, 6                --------------Make the above requested exchange of
                                    words for three lines starting from
                                    the pending line and print the changed
                                    lines on the printer which is lu 6.
```

The changed lines will then be printed on the line printer. If the lu parameter had been omitted, the changed lines would have been displayed on the user's terminal.

A-24 N (DISPLAY A PENDING LINE NUMBER)

Displays the line number of the pending line in the source work area.

- -

N

- -

No parameters required

- -

EXAMPLE:

FILEA contains the following three lines of text,

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode and line one is the pending line then you could give the following commands at your terminal:

```
/P                 --------------Display pending line.
  LINE ONE.        --------------Pending line displayed.
/N                 --------------Display pending line number.
  1                --------------Pending line number.
/+2                --------------Space down two lines and display
                                 pending line.
 LINE THREE.       --------------Pending line displayed.
/N                 --------------Display pending line number.
  3                --------------Pending line number.
```

COMMENTS:

For an explanation of what the source work area is see the EDITR WORK AREAS section in this Section.

A-25 ND (DISPLAY NUMBER IN DESTINATION WORK AREA)

Display the line number of the last line written to the destination work area. The source work area pending line and EOF's are not included in the count.

- -

ND

- -

No parameters required

- -

EXAMPLE:

FILEA contains the following three lines of text

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode and line one is the pending line then you could input the following commands:

```
/P              --------------Display pending line.
  LINE ONE.     --------------Pending line displayed.
/ND             --------------Display line number in destination work area.
                --------------Since no lines have been sent to the
                              destination work area yet, no line number is
                              displayed.
//              --------------Space down one line and display pending line.
  LINE TWO.      --------------Displayed pending line.
/ND             --------------Display line number in destination work area.
  1             --------------Line number of last line written to the
                              destination work area.
/+3             --------------Space down three lines and display pending
                              line.
EOF             --------------End of file.
/ND             --------------Display line number in destination work area.
  3             --------------Line number of last line written to the
                              destination work area.
                              (Remember that the file contains only three
                              lines of text, all three lines have been
                              transferred to the destination work area, and
                              the EOF is not included in the count).
```

COMMENTS:

For a description of the EDITR source and destination work areas see the EDITR WORK AREAS section in this Section.

A-26 H (DISPLAY NUMBER OF CHARACTERS IN PENDING LINE)

Displays the number of characters in the pending line. The count includes a padded blank if the number of characters is uneven.

- -

H

- -

No parameters required

- -

EXAMPLE:

FILEA contains the following two lines of text

```
THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
THIS LINE CONTAINS FORTY-SIX ASCII CHARACTERS.
```

If you are in the edit mode then you could input the following commands at your terminal:

```
/P                  --------------Display pending line.
  THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
/H                  --------------Display number of characters in pending
                                  line.
  42                --------------Number of characters in pending line
                                  (includes padded blank).
//                  --------------Space down one line and display pending
                                  line.
  THIS LINE CONTAINS FORTY-SIX ASCII CHARACTERS.
/H                  --------------Display number of characters in pending
                                  line.
  46                --------------Number of characters in pending line.
```

COMMENTS:

The count includes a padded blank if the number of characters is uneven, because EDITR generates word-length records of two characters per word. This padding may be lost during character edits but replaced as the line is added to the destination work area as shown in the following example.

```
:RU, EDITR    -------------FMGR Command to run EDITR.
SOURCE FILE?  -------------EDITR requesting name of file to be edited.
/0            -------------Put empty file into EDITR's source work area.
```

DISPLAY NUMBER OF CHARACTERS IN PENDING LINE (Continued)

The following command enters a line of text into the EDITR's source work area

```
/ THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
```

The line has not been transferred to the EDITR's destination work area yet; therefore, if you request the number of characters in the pending line the padded blank will not be included.

```
/H                           -------------Display number of characters in
                                          pending line.
 41                          -------------Displayed number of characters.
```

Now by moving the line to the EDITR's destination work area, the padded blank will be included when the number of characters is requested.

```
/1                           -------------Display first line of text and make
                                          it the pending line.
 THIS LINE HAS FORTY-ONE ASCII CHARACTERS.
/H                           -------------Display number of characters in
                                          pending line.
 42                          -------------Displayed number of characters.
```

A-27 HL (DISPLAY HEADER)

Displays a header which facilitates column location.

- -

HL

- -

No parameters required

- -

EXAMPLE:

```
  /HL                  --------------Display column header.
''''/''''1''''/''''2''''/''''3''''/''''4''''/''''5''''/''''6''''/''''7''''/''''8
```

A-28 ∧ (BACK UP IN DESTINATION WORK AREA)

Allows user to back up a specified number of lines in the destination work area and display the new pending line.

- -

∧[n]

- -

n
Number of lines to be backed up in destination work area. Default is to back up one line.

- -

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, the following commands can be input

```
/P                    --------------Display pending line.
  LINE ONE.           --------------Pending line displayed.
/+2                   --------------Advance pending line two lines and
                                    display new pending line.
  LINE THREE.         --------------Pending line displayed.
/∧2                   --------------Back up pending line two lines and
                                    display new pending line.
  LINE ONE.           --------------Pending line displayed.
```

COMMENTS:

Following input of this command, EDITR copies the remainder of the source work area (pending line to end of file) into the destination work area. The destination work area then becomes the new source work area with the pending line moved back n lines. All lines prior to the new pending line are then copied into a new destination work area (lines 1 through n-1).

If n is so large that it causes the pointer to back up past the beginning of the destination work area, the error message ?? is displayed and a smaller value should be specified.

A-29 S (DISPLAY APPROXIMATE NUMBER OF WORDS IN DESTINATION WORK AREA)

Prints the approximate number of words in the destination work area.

- -

S

- -

No parameters required

- -

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, you can input the following commands:

```
/P                   -------------Display pending line.
  LINE ONE.          -------------Pending line displayed.
/L3                  -------------Display three lines of text.  Command
  LINE ONE.                       also moves the three lines to the
  LINE TWO.                       destination work area by moving the
  LINE THREE.                     pending line down three lines.
EOF                  -------------End of file.
/S                   -------------Display number of words in destination
                                  work area.
  19                   -------------Number of words in destination work
                                  area.
```

COMMENTS:

This command can be used to determine the need to break up files into smaller segments (pieces) which would result in large paper tapes. For example, a paper tape containing more than 14,000 words exceeds the standard box size. If the word count is determined with the S command, the file can be broken with a series of zero-length records in the appropriate spots.

A-30 P (EDIT AND DISPLAY PENDING LINE)

Edits and displays the pending line.

- -

P [editstring]

- -

editstring
The editstring can be composed of delimiters used as place holders to preserve existing text, new text to be inserted or to replace existing text, and a non-printing control command to replace, insert or delete characters. If no editstring is specified, then the existing pending line is displayed. (Delimiters must be the current prompt character-default prompt character is a slash).

- -

EXAMPLE:

FILEA contains the following two lines of text:

```
LINE ONE.
LINE TWO.
```

The following edit session demonstrates use of the "P" command.

```
:RU,EDITR          -------------FMGR Command to run EDITR.
SOURCE FILE?
/FILEA             -------------File to be edited.
  LINE ONE.        -------------EDITR responds with first line of file.
/P                 -------------Display pending line.
  LINE ONE.        -------------Pending line displayed.
/P/////FIRST.      -------------Edit pending line and display new
                                pending line.
  LINE FIRST.      -------------EDITR responds with new pending line.
/ER                -------------End edit session replacing old
                                version of FILEA with edited version.
END OF EDIT        -------------EDITR response indicating edit session
                                is over.
```

COMMENTS:

The O and C commands are similar to the P command when used for character edits (see the following two sections for a description of these commands).

The P command is also used frequently to display the unedited pending line. Many of the examples for other commands make use of this feature.

A-31 C (EDIT PENDING LINE AND ADVANCE LINE)

Allows editing of the pending line and advances the pending line following the edit.

- -

```
C[editstring]
```

- -

editstring
The editstring can be composed of delimiters used as place holders to preserve existing text, new text to be inserted or to replace existing text, and a non-printing control command to replace, insert or delete characters. If no editstring is specified, then the pending line will be displayed, no change will be made to it, and the pending line will be advanced one line and displayed. (Delimiters must be the current prompt character-default prompts character is a slash).

- -

EXAMPLE:

FILEA contains the following two lines of text:

```
LINE ONE
LINE TWO
```

To change the above two lines, the following set of commands can be input while in the edit mode

```
/P                  ------------Display pending line.
  LINE ONE          ------------Pending line displayed.
/C/////FIRST        ------------Edit pending line changing ONE to FIRST.
                                Display edited line, advance pending
                                line, and display new pending line.
  LINE FIRST        ------------Edited line displayed.
  LINE TWO          ------------New pending line.
/C/////SECOND       ------------Edit pending line changing TWO to
                                SECOND.  Display edited line, advance
                                pending line, and display new pending
                                line.
  LINE SECOND       ------------Edited line displayed.
EOF                 ------------End of file.
```

A-32 O (DUPLICATE PENDING LINE AND EDIT)

Duplicates the pending line and allows character edits on the duplicate.

O[editstring]

---

editstring
The editstring is composed of delimiters used as place holders to preserve existing text, new text to be inserted or to replace existing text, and a non-printing control command to replace, insert or delete characters. If no editstring is specified, then the existing pending line is duplicated and displayed. (Delimiters must be the current prompt character-default prompt character is a slash).

EXAMPLE:

FILEA contains the single line of text:

```
LINE ONE
```

If you are in the edit mode, the following EDITR commands can be input at your terminal to create two additional lines:

```
/O                  ------------Duplicate pending line and display
                                duplicate.
  LINE ONE          ------------Duplicate line displayed.
/P/////TWO          ------------Change ONE to TWO and display edited
                                line.
  LINE TWO          ------------Edited line displayed.
/O                  ------------Duplicate pending line and display
                                duplicate.
  LINE TWO          ------------Duplicate line displayed.
/P/////THREE        ------------Change TWO to THREE and display edited
  LINE THREE                    line.
/1                  ------------Make line one pending line and display it.
  LINE ONE          ------------Pending line displayed.
/L3                 ------------Display three lines of text.
  LINE ONE
  LINE TWO
  LINE THREE
EOF                 ------------End of file.
```

COMMENTS:

A more efficient method of accomplishing the above example would be to use the editstring parameter in the O command. For example:

```
/P                    -----------Display pending line.
  LINE ONE            -----------Pending line displayed.
/O/////TWO            -----------Duplicates pending line
                                 and edits duplicate.
  LINE TWO            -----------Edited line displayed and
                                 made new pending line.
/O/////THREE          -----------Duplicates pending line and
                                 edits duplicate.
  LINE THREE          -----------Edited line displayed and made
                                 new pending line.
```

A-33 R (REPLACE PENDING LINE WITH TEXT)

Replaces the pending line with new text entered at the terminal.

- -

```
Rtext
```

- -

text
New line of text to replace pending line. If no new text is specified, the pending line becomes zero length.

- -

EXAMPLE:

The following example demonstrates use of this command to replace the pending line with a new line of text.

```
/P                    ------------Display pending line.
  LINE ONE.           ------------Pending line displayed.
/RFIRST LINE.         ------------Replace pending line text with new
                                  text.
/P                    ------------Display pending line.
  FIRST LINE.         ------------Pending line displayed.
```

A-34 I (INSERT TEXT BEFORE PENDING LINE)

Inserts a new line of text before the pending line.

```
Itext
```

---

text
Line of text to be inserted before pending line in destination work area. If command is entered with no new text, the inserted line becomes zero length.

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, you can input the following commands:

```
/P            -------------Display pending line.
  LINE ONE.   -------------Pending line displayed.
//            -------------Advance pending line one line and
                           display new pending line.
  LINE TWO.   -------------Pending line displayed.
/INEW LINE.   -------------Insert new line of text before
                           pending line.
/1            -------------Make line one pending line and display
                           pending line.
  LINE ONE.   -------------Pending line displayed.
/L4           -------------Display four lines of text.
  LINE ONE.
  NEW LINE.
  LINE TWO.
  LINE THREE.
EOF
```

A-35 ( ) (INSERT TEXT AFTER PENDING LINE)

Used to insert new text immediately after the pending line. The new line then becomes the pending line. Note that ( ) represents a space.

- -

```
( ) text
```

- -

text
New line of text to be inserted following pending line. If command is entered without new text, the new line has zero length.

- -

EXAMPLE:

FILEA contains the following three lines of text:

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, then you could input the following commands:

```
/P              ------------Display pending line.
  LINE ONE.     ------------Pending line displayed.
/ NEW LINE.     ------------Insert new line of text following
                            pending line and make it new pending
                            line.
/P              ------------Display pending line.
  NEW LINE.     ------------Pending line displayed.
/1              ------------Make line one pending line and display
                            pending line.
  LINE ONE.     ------------Pending line displayed.
/L4             ------------Display four lines of text starting from
                            pending line.
  LINE ONE.
  NEW LINE.
  LINE TWO.
  LINE THREE.
EOF             ------------End of file.
```

A-36 - (DELETE A NUMBER OF LINES)

Deletes a specified number of lines in the text.

- -

-n

- -

n
Number of lines of text to be deleted (starting with pending line).
If no value is specified, one line is deleted.

- -

EXAMPLE:

FILEA contains the following three lines of text

```
LINE ONE.
LINE TWO.
LINE THREE.
```

If you are in the edit mode, then you could give the following set of commands:

```
/P              -----------Display pending line.
  LINE ONE.     -----------Pending line displayed.
/-2             -----------Delete two lines of text and display
                           new pending line.
  LINE THREE.   -----------Pending line displayed.
/1              -----------Make line one pending line and display
                           pending line.
  LINE THREE.   -----------Pending line displayed.
/L3             -----------Display three lines of text.
  LINE THREE.
EOF             -----------End of file.
```

A-37 CNTL/R REPLACE CHARACTERS

Used in character edits to replace characters on a line.

CNTL/R

A control R is input by striking the "R" key while depressing the "CNTL" key on the terminal. It is a non-printing character.

(Refer to the EDITR IN A MULTIPOINT ENVIRONMENT section for Multipoint operation).

COMMENTS:

Character replacement is the EDITR default mode so that the control character CNTL/R can be omitted when EDITR is initially turned on, or following a P,C or O command. It is only required following one of the control characters CNTL/I, CNTL/S, or CNTL/C.

Any embedded character you wish to preserve in the original text is skipped by entering the current delimiter (/). Using the tab character will also cause characters to be skipped. Skipped characters in the new line appear exactly as they did in the old line. A carriage return preserves the rest of the line unless CNTL/T was specified to truncate.

An example of a character edit combination where the CNTL/R is required is shown below.

```
/P                 --------------Display pending line.
  OLD EDITR TEST FILE
/CNEW//////         /////FOR TESTS

                       CNTL/S  ----Insert new text.

                 CNTL/R ----------Return to replace with control
                                  character to preserve text.

         CNTL/C ----------------Cancel with space for place holder.

 ------------------------------Replace with no control character.

 NEW EDITR FILE FOR TESTS -----Edited line of text.
```

A-38 CNTL/I OR CNTL/S (INSERT CHARACTERS)

Used to insert characters in the pending line.

- -

```
CNTL/I
 or
CNTL/S
```

- -

A control I or S is input by striking the "I" or "S" key while depressing the "CNTL" key on the terminal. They are non-printing characters.

(Refer to the EDITR IN A MULTIPOINT ENVIRONMENT section for Multipoint operation).

- -

EXAMPLES:

1) CNTL/S example.

```
/P                   ---------------Display pending line.
  LINE ONE.          ---------------Pending line displayed.
/P//////// OF TEXT  --------------Edit pending line and display it.
          ↑
          └- CNTL/S

  LINE ONE OF TEXT.  ------------Edited version of pending line.
```

2) CNTL/I example.

```
/P                   ---------------Display pending line.
  LINE ONE.          ---------------Pending line displayed.
/P////////↑              -----------Edit pending line and display it.
          │
          └CNTL/I
 OF TEXT                 -----------CNTL/I also acts as a carriage return.

 LINE ONE OF TEXT.       -----------Edited version of pending line.
```

COMMENTS:

The CNTL/I and CNTL/S commands are used with the P,C, and O commands (for a description of these commands, see the explanations given earlier in this Section).

Once CNTL/I or CNTL/S is entered, entering the current delimiter (/) to do character skipping inserts blanks. The tab character also causes the tabbed number of blanks to be inserted.

A-39 CNTL/C (CANCEL CHARACTERS)

Used to delete characters from the pending line.

- -

CNTL/C

- -

A control C is input by striking the "C" key while depressing the "CNTL" key on the terminal. It is a non-printing character.

(Refer to the EDITR IN A MULTIPOINT ENVIRONMENT section for Multipoint operation).

- -

EXAMPLE:

```
/P                  --------------Display pending line.
  LINE ONE OF TEXT. --------------Pending line displayed.
/P////////XXXXXXXX  --------------Edit pending line and display.
          ^
          |
          CNTL/C    --------------Delete characters.

  LINE ONE.            -----------Edited version of pending line.
```

COMMENTS:

Each character or place holder entered after CNTL/C will delete one character from the pending line. Character skipping, i.e., entering the current delimiter deletes characters. The tab character deletes everything up to the tab stop. When carriage return is entered, the pending line will be left justified.

The CNTL/C command is used in conjunction with the P,C, or O commands. For a description of these commands, see the explanations given earlier in this Section.

A-40 CNTL/T (TRUNCATE CHARACTERS)

Used to delete characters from the end of the pending line. When the control character is entered, the remainder of the line is eliminated.

- -

```
CNTL/T
```

- -

A control T is input by striking the "T" key while depressing the "CNTL" key on the terminal. It is a non-printing character.

- -

EXAMPLE:

```
/P                   -------------Display pending line.
  LINE ONE OF TEXT       ---------Pending line displayed.
/P////////^          -------------Edit pending line and display.
          |
       CNTL/T        -------------Eliminate characters following CNTL/T.

  LINE ONE           -------------Pending line displayed.
```

COMMENTS:

When operating under a multipoint, the CNTL/T command can be replaced by the Q command. For details on the Q command, see the section on EDITR IN A MULTIPOINT ENVIRONMENT.

A-41 B (FIND A LINE WITH A FIND FIELD - SOF TO EOF)

Searches the source work area, from the start-of-file to the end-of-file, for the first line of text which matches the find field. When a line containing the correct character string is found, it becomes the new pending line. The search ends at EOF if no match is found.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  B^            --------------Search for zero length record.
  └---CNTL/@

  Btext         --------------Search for left justified text.

  Btext         --------------Search for text anywhere in line.
  ↑
  '---ESC key

  B/text        --------------Search for text in window.

  B;text        --------------Search for field at tab stop.

  .B            --------------Successive search for same field.  The same
                              line will always be found by this command.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

CNTL/@
A control @ is input by striking the "@" key while depressing the "CNTL" key on the terminal. It is a non-printing character.

text
This is the portion of text that is to be found by the B command.

ESC
The ESC key is located on the terminal. It is a non-printing character.

- -

EXAMPLES:

FILEA contains the following four lines of text

```
  LINE ONE OF EXAMPLE.
  LINE TWO OF EXAMPLE.
                          ---(zero length record)
         LINE FOUR OF EXAMPLE.
```

A-41 FIND A LINE WITH A FIND FIELD (Continued)

1) B CNTL/@ example.

```
/B                    -------------Search for zero length record and make it
 ↑                                 pending line.
--CNTL/@              -------------Pending line (zero length record).
/∧                    -------------Move pending line back one line and
                                   display new pending line.
 LINE TWO OF EXAMPLE.      --------Pending line displayed.
```

2) Btext example.

```
/BLINE                -------------Search for first occurrence of LINE (left
                                   justified), making line pending line, and
                                   display it.
 LINE ONE OF EXAMPLE.      --------Pending line displayed.
```

3) Btext example.

```
↑
 ---ESC

/BTWO                 -------------Search for first occurrence of TWO,
                                   making line containing TWO pending line,
   --ESC                           and display it.
 LINE TWO OF EXAMPLE.       -------Pending line displayed.
```

4) B/text example.

```
/W6, 11               -------------Make window between columns 6 through 11.
B/LINE                -------------Find first occurrence of LINE within
                                   window constraints, making line
                                   containing text pending line, and display
                                   pending line.
        LINE FOUR OF EXAMPLE.
```

5) B;text example

```
/B;LINE               -------------Find first occurrence of LINE at tab
                                   stop, making it pending line and
                                   display pending line.
        LINE FOUR OF EXAMPLE.
```

6) B example.

```
/B;LINE               -------------Find first occurrence of LINE at tab
                                   stop, making it pending line and
                                   display pending line.
        LINE FOUR OF EXAMPLE.
/1                    -------------Make line one pending line and display
                                   pending line.
 LINE ONE OF EXAMPLE.
/B                    -------------Search for same field sought previously.
        LINE FOUR OF EXAMPLE.
```

A-41 FIND A LINE WITH A FIND FIELD (Continued)

COMMENTS:

The B command always starts at the first line of the file to find the text. Once the text is found, the line containing it becomes the pending line. The B command cannot be used for successive searches for the same text, because it will always find the same line unless the text being searched for is changed on that line. All lines passed over in the search are written to the destination work area; they are not deleted. The B command always forces the destination work area to replace the source work area before the search.

The slash in the command format B/text, is representative of the initial delimiter. If the delimiter has been changed (X-CHANGE EDITR PROMPT CHARACTER), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (T - SET TAB STOPS), the new tab character must be used in place of the ;.

Note, that if the window is set at its default values (1 to 150), the B/text command does the same thing as the Btext.

↑--ESC

A-42 F (FIND A LINE WITH A FIND FIELD - PENDING LINE TO EOF)

Causes a search of the source work area, beginning at the line following the pending line and continuing to the end-of-file for a line containing a specified character string. When a line containing this field is found, it becomes the new pending line. If the line is not found, the search ends at the end-of-file.

- -

```
Ftext      --------------Search for left justified field.

Ftext      --------------Search for field embedded anywhere in line.
↑--ESC

F/text     --------------Search for field within a window.

F;text     --------------Search for field beginning at tab stop.

F          --------------Successive searches for same field.
```

- -

text
This is the portion of text that is to be found by the F command.

ESC
The ESC key is located on the terminal. It is a non-printing character.

- -

A-42 F (FIND A LINE WITH A FIND FIELD - PENDING LINE TO EOF) (CONT'D.)

EXAMPLE:

FILEA contains the following four lines of text

```
THIS IS AN EXAMPLE
FOR DEMONSTRATING
THE F COMMAND
      THE F COMMAND
```

1) Ftext example.

```
/P                    --------Display pending line.
  THIS IS AN EXAMPLE  --------Pending line displayed.
/FTH                  --------Search for left justified field
                              with text TH, making line containing
                              field pending line, and display it.
  THE F COMMAND       --------Pending line displayed.
/2                    --------Advance pending line one line and display
                              new pending line.

  FOR DEMONSTRATING   --------New pending line displayed.
/FTH                  --------Search for left justified field
                              with text TH, making line containing
                              field pending line, and display it.
  THE F COMMAND       --------Pending line displayed.
```

2) Ftext example.

```
 ↑
 └--ESC
```

```
/P                    --------Display pending line.
  THIS IS AN EXAMPLE  --------Pending line displayed.
/FDEMO                --------Search for text embedded anywhere
  ↑                           in line, making line pending line,
  └--ESC                      and display it.

  FOR DEMONSTRATING   --------Pending line containing text.
/+                    --------Advance pending line by one line
                              and display new pending line.
  THE F COMMAND       --------Pending line displayed.
/FDEMO                --------Search for text embedded anywhere
  ↑                           in line, making line pending line,
  └--ESC                      and display it.

EOF                   --------End of file.  Indicates remainder of
                              file was searched and text was not found.
```

A-42 <u>(F FIND A LINE WITH A FIND FIELD (CONT'D.)</u>

3) F/text example.

```
/P                           --------Display pending line.
  THIS IS AN EXAMPLE         --------Pending line displayed.
/W6,10                       --------Set window between columns 6 and 10.
/F/COM                       --------Search for text within window, making
                                     line containing text pending line,
                                     and display pending line.
  THE F COMMAND             --------Pending line displayed.
```

4) F;text example.

```
/P                           --------Display pending line.
  THIS IS AN EXAMPLE         --------Pending line displayed.
/F;THE                       --------Search for text beginning at tab
                                     stop, making line containing text
                                     pending line, and display pending
                                     line.
          THE F COMMAND      --------Pending line displayed.
```

5) F example.

```
/P                          -------Display pending line.
  THIS IS AN EXAMPLE        -------Pending line displayed.
/FTHE                       -------Search for text embedded anywhere in line,
  ↑                                making line containing text new pending
  --ESC                            line, and display new pending line.

  THE F COMMAND             -------Pending line displayed.
/F                          -------Successive search for same text,
                                   making line containing text new
                                   pending line, and display line.
         THE F COMMAND        -----Pending Line displayed.
```

COMMENTS:

The slash in the command format F/text, is representative of the initial delimiter. If the delimiter has been changed (see the CHANGE EDITR PROMPT CHARACTER section), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (see the SET TAB STOPS section), the new tab character must be used in place of the ;.

Note that if the window is set at its default values (1 to 150), then the F/text command does the same thing as the Ftext.
↑--ESC

A-43 D (DELETE LINES TO FIND FIELD OR EOF)

Deletes a block of text from the pending line to the line containing a specified field. After the deletion, the line of text containing the specified field becomes the new pending line. If the specified field is not encountered, the remainder of the file is deleted. If this command is used without specifying a field, the last field entered is used.

```
D                  --------Delete lines until a zero length field is
↑                          encountered.
|--CNTL/@

Dtext              --------Delete lines until specified text is found
                           in left justified field.

Dtext              --------Delete lines until specified text is found in
 ↑--ESC                    field located anywhere in line.

D/text             --------Delete lines until specified text is found in
                           field located within window.

D;text             --------Delete lines until specified text is found in
                           field beginning at tab stop.

D                  --------Successive deletes to specified text in prior
                           delete command.
```

PARAMETERS:
CNTL/@
A control @ is input by striking the "@" key while depressing the "CNTL" key on the terminal. It is a non-printing character.

text
This is the portion of text that is to be found by the D command.

ESC
The ESC key is located on the terminal. It is a non-printing character.

EXAMPLE:

FILEA contains the following 7 lines of text

```
THIS IS AN EXAMPLE
DEMONSTRATING THE
                    --------Zero length record.
DELETE LINES COMMAND.
                    --------Zero length record.
THIS NEXT LINE BEGINS
      IN COLUMN 7.
```

A-43 <u>DELETE LINES (Continued)</u>

```
1)  D  example.
     ↑
     └-CNTL/@

    /1                     --------Make line one pending line and display it.
      THIS IS AN EXAMPLE   ----Pending line displayed.
    /D                     --------Delete all lines until zero length record is
      ↑                            found.  Make zero length record pending line
      └-CNTL/@                     and display it.

                           --------Zero length record displayed.
   /1                      --------Make line one pending line and display it.
                           --------Pending line displayed (zero length record).
   /L6                     --------Display six lines of text.

       DELETE LINES COMMAND.

       THIS NEXT LINE BEGINS
             IN COLUMN 7.
   EOF                      -------End of file.

2)  Dtext example.

    /1                     --------Make line one pending line and display it.
      THIS IS AN EXAMPLE   ----Pending line displayed.
    /DTHIS                 --------Delete all lines until THIS is encountered
                                   in a left justified field, making line
                                   containing text pending line, and displaying it.
      THIS NEXT LINE BEGINS  --Pending line displayed.
    /1                     --------Make line one pending line and display it.
      THIS NEXT LINE BEGINS  --Pending line displayed.
    /L6                    --------Display six lines of text.
      THIS NEXT LINE BEGINS
            IN COLUMN 7.
    EOF                    --------End of file.

3)  Dtext example.
    ↑
    └--ESC
      /1                   --------Make line one pending line and display it.
        THIS IS AN EXAMPLE ---Pending line displayed.
      /DBEGINS             --------Delete all lines until BEGINS is
        ↑                          encountered anywhere in file, making
      --ESC                        line containing BEGINS pending line,
                                   and displaying it.
        THIS NEXT LINE BEGINS--Pending line displayed.
      /1                    -------Make line one pending line and display it.
        THIS NEXT LINE BEGINS--Pending line displayed.
      /L6
        THIS NEXT LINE BEGINS
                IN COLUMN 7.
      EOF                    ------End of file.
```

A-43 D (DELETE LINES CONTINUED)

4) D/text example.

```
/1                          --------Make line one pending line and display it.
  THIS IS AN EXAMPLE            ----Pending line displayed.
/W10,15                     --------Set window between columns 10 and 15.
/D/LINE                     --------Delete all lines until LINE is located in
                                    a field within the specified window.
                                    Make line containing LINE pending line and
                                    display it.
  THIS NEXT LINE BEGINS        -----Pending line displayed.
/1                           -------Make line one pending line and display
                                    it.
  THIS NEXT LINE BEGINS          ---Pending line displayed.
/L6                          -------Display six lines of text.
  THIS NEXT LINE BEGINS
        IN COLUMN 7.
EOF                         --------End of file.
```

5) D;text example.

```
/1                          --------Make line one pending line and display it.
  THIS IS AN EXAMPLE          ------Pending line displayed.
/D;IN                       --------Delete all lines until IN is encountered
                                    at the first tab stop on some line.  Make
                                    line containing IN pending line and
                                    display it.
        IN COLUMN 7.           -----Pending line displayed.
/1                           -------Make line one pending line and display it.
        IN COLUMN 7.           -----Pending line displayed.
/L6                          -------Display six lines of text.
        IN COLUMN 7.
EOF                          -------End of file.
```

6) D example.

```
/1                          --------Make line one pending line and display it.
  THIS IS AN EXAMPLE           -----Pending line displayed.
/D                          --------Delete all lines until a zero length
  ↑                                 record is encountered, making zero
  └--CNTL/@                         length record pending line and display it.
                            --------Zero length record displayed as pending
                                    line.
/L6                         --------Display six lines of text.

  DELETE LINES COMMAND

  THIS NEXT LINE BEGINS
        IN COLUMN 7.
```

A-43 (DELETE LINES CONTINUED)

```
EOF                       --------End of file.
/1                        --------Make line one pending line and display it.
                             -----Pending line displayed (zero length
                                  record).
/D                        --------Delete all lines until text field
                                  specified in last delete command is
                                  encountered.  Make line containing
                                  text new pending line and display it.
                             -----Pending line displayed (zero length
                                  record).
/L6                       --------Display six lines of text.

  THIS NEXT LINE BEGINS
        IN COLUMN 7.
EOF                       --------End of file.
```

COMMENTS:

The slash in the command for D/text, is representative of the initial delimiter. If the delimiter has been changed (see the CHANGE EDITR PROMPT CHARACTER section), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (see the SET TAB STOPS section), the new tab character must be used in place of the ;.

Note that if the window is set at its default values (1 to 150), then the

D/text

command does the same thing as the

```
Dtext.
↑
└--ESC
```

A-44 J (JUMP TO FIND FIELD LINE)

Used to either delete or move text. It causes the pending line pointer to jump to a line containing a specified text string in the source work area without moving the jumped over lines to the destination work area.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

  Jtext                       ------------Jump to line containing text in left
                                          justified field.

  Jtext                       ------------Jump to line containing text embedded
   ↑                                      anywhere in line.
   --ESC

  J/text                      ------------Jump to line containing text embedded
                                          anywhere within window.

  J                           ------------Jump to same line that was jumped to
                                          with prior jump command.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

text
This is the portion of text that is to be found by the J command.

ESC
The ESC key is located on the terminal. It is a non-printing character.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

EXAMPLE:

```
Source Work Area            Pending Line           Destination Work Area
----------------            ------------           ---------------------
FIRST LINE                   ----PL
      SECOND LINE
THIRD LINE
      FOURTH LINE

/JTHIRD   ------------------------------Jump to line with text in left
                                        justified field.

Source Work Area            Pending Line           Destination Work Area
----------------            ------------           ---------------------
FIRST LINE                                         FIRST LINE
      SECOND LINE
THIRD LINE                   ----PL
      FOURTH LINE
```

A-44 J (JUMP TO FIND FIELD LINE CONTINUED)

```
/JCOND      ----------------------------Jump to line with text embedded in
  ↑                                     line.
  └-ESC

Source Work Area           Pending Line            Destination Work Area
----------------           ------------            ---------------------
FIRST LINE                                         FIRST LINE
      SECOND LINE           ----PL                 THIRD LINE
THIRD LINE
      FOURTH LINE

/W10,20     ----------------------------Set window between columns 10 to 20.
/J/LINE     ----------------------------Jump to line containing first
                                        occurrence of text which is embedded
                                        with window.

Source Work Area           Pending Line            Destination Work Area
----------------           ------------            ---------------------
FIRST LINE                                         FIRST LINE
      SECOND LINE           ----PL                 THIRD LINE
THIRD LINE                                               SECOND LINE
      FOURTH LINE

/J          ----------------------------Jump to same line as previous jump
                                        command.

Source Work Area           Pending Line            Destination Work Area
----------------           ------------            ---------------------
FIRST LINE                                         FIRST LINE
      SECOND LINE           ----PL                 THIRD LINE
THIRD LINE                                               SECOND LINE
      FOURTH LINE                                        SECOND LINE

/1          ----------------------------Make line one pending line.  Also,
                                        copies remainder of source work area
                                        from pending line on down into
                                        destination work area and then makes
                                        destination work area new source work
                                        area.

Source Work Area           Pending Line            Destination Work Area
----------------           ------------            ---------------------
FIRST LINE                  ----PL
THIRD LINE
      SECOND LINE
      SECOND LINE
      SECOND LINE
THIRD LINE
      FOURTH LINE
```

A-44 J (JUMP TO FIND FIELD LINE)

COMMENTS:

Before using the J command for deleting and moving blocks of text, you should be familiar with the concept of source and destination work areas (refer to the EDITR WORK AREAS). Remember that whenever the pending line pointer goes back to a prior line in the source work area, the destination work area will become the new source work area.

Whenever text is sought with a J command, the first line containing the text in the source work area will become the pending line regardless of where the pending line pointer was prior to the command. The only line moved by the J command to the destination work area is the pending line prior to the jump.

The slash in the command format J/text, is representative of the initial delimiter. If the delimiter has been changed (see the CHANGE EDITR PROMPT CHARACTER section), then the appropriate delimiter character must be used in place of the /. If the initial tab character is changed (see the SET TAB STOPS section), the new character must be used in place of the ;.

Note that if the window default values (1 to 150) are used, the

```
J/text
```

command does the same thing as the

```
Jtext
 ↑
 --ESC
```

A-45 G (CHARACTER REPLACE ON PENDING LINE)

Exchanges an old character string with a new character string on the pending line. The new pending line following the exchange is then displayed.

- -

```
Goldstring/newstring
```

- -

oldstring
Old character string to be exchanged with new. Can be located anywhere in pending line. If several identical character strings are located on the line, all of them will be exchanged for the new string. Can be any length except zero.

newstring
This is the new character string which is to replace the old string. Can be any length including zero.

- -

A-45 G (CHARACTER REPLACE ON PENDING LINE CONTINUED)

EXAMPLE:

```
/P              ---------------Display pending line.
  OLD STRING           -------Pending line displayed.
/GOLD/NEW       ---------------Exchange OLD text for NEW text and display
                               new pending line.
  NEW STRING           -------Pending line displayed.
```

COMMENTS:

If the pending line contains several identical strings of text, but not all of them are desired to be changed, then the set window command can be used to selectively change character strings. For example, if the pending line is

```
OLD OLD OLD OLD
```

and only the third occurrence of OLD is desired to be exchanged for NEW, the following commands can be input to accomplish this.

```
/W8,12         ---------------Set window between columns 8 and 12.
/GOLD/NEW      ---------------Exchange OLD text with NEW text and
                              display new pending line.
  OLD OLD NEW OLD      -------Pending line displayed.
```

Once the Goldstring/newstring Command has been issued, subsequent exchanges may be made by specifying G without any parameters, i.e., G parameters remain in effect until changed.

The exchange fields used in the G command are not tabbed, so that the tab character can be used like any other character in the exchange string.

The only time the current delimiter is recognized as a delimiter is the first time it occurs separating the old string and the new string.

A-46 Y (EXCHANGE PENDING LINE, DISPLAY NEXT OCCURRENCE OF PATTERN)

Performs an exchange of an old string with a new string on the pending line.

- -

```
Yoldstring/newstring  ------Exchange all occurrences of old string on
                            pending line with new string.  After exchange,
                            the next occurrence of old string becomes the
                            new pending line.  If same exchange on new
                            pending line is desired, only the command Y need
                            be input.
```

- -

A-46 Y (EXCHANGE ON PENDING LINE CONTINUED)

- -

oldstring
Old string of text to be exchanged with new text string. It can be any length except zero.

newstring
This is the new text string which is to replace the old. It can be any length including zero.

- -

EXAMPLE:

FILEA contains the following five lines of text

```
THIS IS AN OLD STRING.
THE OLD STRING CAN BE
REPLACED BY A NEW
STRING WITH THE Y
COMMAND.
```

To change all occurrences of the text, STRING, with the text, TEXT, the following commands can be input while in the edit mode.

```
/YSTRING/TEXT         -----------Exchange STRING with TEXT on pending
                                 line, display edited line, find next
                                 occurrence of STRING, making it the
                                 pending line and display it.
 THIS IS AN OLD TEXT.      -----Edited line displayed.
 THE OLD STRING CAN BE     -----New pending line displayed.
/Y             -------------------Exchange STRING with TEXT on pending
                                 line, display edited line, find next
                                 occurrence of STRING, making it
                                 pending line and display it.
 THE OLD TEXT CAN BE       -----Edited line displayed.
 STRING WITH THE Y         -----New pending line displayed.
/Y              ------------------Exchange STRING with TEXT on pending
                                 line, display edited line, find next
                                 occurrence of STRING, making it pending
                                 line and display it.
 TEXT WITH THE Y           -----Edited line displayed.
EOF             ------------------End of file.
```

A-46 Y (EXCHANGE ON PENDING LINE CONTINUED)

COMMENTS:

A combination of Y and F commands can be used to do selective exchanges. The F command is entered (without parameters) to skip an exchange on this occurrence and find the next occurrence. Using FILEA as an example, the first and last occurrence of STRING can be changed to TEXT leaving the second occurrence unchanged by the following commands:

```
/YSTRING/TEXT     -------------Exchange STRING with TEXT on pending line,
                               display edited line, find next occurrence
                               of STRING, making it the pending line and
                               display it.
  THIS IS AN OLD TEXT.    -----Edited line displayed.
  THE OLD STRING CAN BE   -----New pending line displayed.
/F                -------------Find next occurrence of STRING and make it
                               pending line leaving present pending line
                               unchanged.
  STRING WITH THE Y       -----New pending line displayed.
/Y          -------------------Exchange STRING with TEXT on pending line,
                               display edited line, find next occurrence of
                               STRING, making it pending line and display
                               it.
  TEXT WITH THE Y         -----Edited line displayed.
EOF         -------------------End of file.
```

If the string to be changed occurs more than once on the pending line, all occurrences of the string will be replaced by the new string. To selectively change identical strings on the pending line, the set window command can be used. For example, if the pending line were

```
OLD OLD OLD OLD
```

and only the third occurrence of old was desired to be changed, the following commands can be input.

```
/W8,12     -------------------Set window between columns 8 and 12.
/YOLD/NEW  -------------------Exchange OLD with NEW on pending line,
                              display edited line, find next occurrence
                              of old, making it pending line and display
                              it.
   OLD OLD NEW OLD       -----Edited line displayed.
 EOF         -----------------End of file.
```

The exchange fields used in the Y command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter (/) is recognized. It is assumed to separate the old string and the new string.

A-47 X (ENABLE EXCHANGE PATTERN OVER RANGE OF LINES, WITH LIST)

Used to make string exchanges over a user specified range of lines and then list the lines which have been changed. It is always used in combination with a second command which defines the range of lines over which the string exchange is to take place.

- -

Xoldstring/newstring	Exchange all occurrences of old string with new string over a specified range of lines within a column window. All changed lines are listed.
range command	Range of lines over which exchanges are to take place.

- -

oldstring
Old string of text which is to be exchanged with new string. It can be any length except zero.

newstring
This is the new string of text which is to be exchanged for the old. It can be any length including zero.

range command
This is any command which will cause the pending line pointer to advance through the source file.

- -

EXAMPLE:

FILEA contains the following five lines of text:

```
THIS IS AN OLD STRING.
THE OLD STRING CAN BE
REPLACED BY A NEW
STRING WITH THE X
COMMAND.
```

If it is desired to replace all occurrences of the text, STRING, with the text, TEXT, the following commands can be input

/XSTRING/TEXT	------------Exchange all occurrences of STRING with TEXT over the range specified in the next command.

A-47 X ENABLE EXCHANGE PATTERN OVER RANGE OF LINES CONTINUED)

```
/FEOF                   ------------Search all lines for STRING and change
                                    to TEXT until line with EOF is found in
                                    left justified field.
  THIS IS AN OLD TEXT         ------Changed line of text.
  THE OLD TEXT CAN BE         ------Changed line of text.
  TEXT WITH THE X             ------Changed line of text.
EOF                     ------------End of file.
```

COMMENTS:

Other examples of for range commands would be the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n).

The default print device is the terminal that you input the commands at, but by using the delete lines command (D), the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n), the optional LU command can be invoked to send the changed lines to the printer or a disk file.

The exchange takes place over all occurrences of the old string text within each line unless the window command is used to limit the range of columns over which the exchange is to take place. Using FILEA as an example to demonstrate this the following commands can be input to change only the STRING which starts in column 9 of line two.

```
/W8,15                 -----------Set window between columns 8 and 15.
/XSTRING/TEXT          -----------Exchange command.
/+10                   -----------Range command.  Advance pending
                                  line down 10 lines.
  THE OLD TEXT CAN BE       -----Changed line.
EOF                    -----------End of file.
```

If additional exchanges are to be made beyond the line positioned to with the range command, the current parameters of the X command may be re-enabled by specifying X with no parameters.

The exchange fields used in the X command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the old string and the new string.

A-48 Z (ENABLE EXCHANGE PATTERN OVER RANGE OF LINES, WITHOUT LIST)

Used to exchange a string of text with a new string of text over a specified number of lines. It is always used in combination with a second command which specifies the range of lines over which the exchange is to take place. It is the same as the X command except that there is no listing of the changed lines.

Zoldstring/newstring	------Exchange all occurrences of old string with new string over a specified range of lines within a column window.
range command	------Range of lines over which exchanges are to take place.

---

oldstring
Old string of text which is to be exchanged with new string. It can be any length except zero.

newstring
This is the new string of text which is to be exchanged for the old. It can be any length including zero.

range command
This is any command which will cause the pending line pointer to advance through the source file.

EXAMPLE:

FILEA contains the following five lines of text

```
THIS IS AN OLD STRING.
THE OLD STRING CAN BE
REPLACED BY A NEW
STRING WITH THE Z
COMMAND.
```

If it is desired to replace all occurrences of the text, STRING, with the text, TEXT, the following commands can be input

/ZSTRING/TEXT	----------Exchange all occurrences of STRING with TEXT over the range specified in the next command.

A-48 Z (ENABLE EXCHANGE PATTERN OVER RANGE OF LINES, WITHOUT LIST) (Cont'd.)

```
/FEOF              -----------Search all lines for STRING and
                              change to TEXT until line with
                              EOF is found in left justified
                              field.

EOF                -----------End of file.
/1                            Make line one pending line.
/L5                           Display five lines of text.
  THIS IS AN OLD TEXT.
  THE OLD TEXT CAN BE
  REPLACED BY A NEW
  TEXT WITH THE Z
  COMMAND.
EOF                -----------End of file.
```

COMMENTS:

Other examples for range commands would be the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n).

The exchange takes place over all occurrences of the old string text within each line unless the window command is used to limit the range of columns over which the exchange is to take place. Using FILEA as an example to demonstrate this, the following commands can be input to change only the STRING which starts in column 9 of line two.

```
/W8,15              ----------Set window between columns 8 and 15.
/ZSTRING/TEXT       ----------Exchange command.
/+10                ----------Range command. Advance pending line
                              down 10 lines.
EOF                 ----------End of file.
/1                  ----------Make line one pending line.
/L5                 ----------Display five lines of text.
  THIS IS AN OLD STRING.
  THE OLD TEXT CAN BE
  REPLACED BY A NEW
  STRING WITH THE Z
  COMMAND.
EOF                 ----------End of file.
```

A-48 Z (ENABLE EXCHANGE PATTERN OVER RANGE OF LINES, WITHOUT LIST) (Cont'd.)

If additional exchanges are to be made beyond the line positioned to with the range command, the current parameters of the Z command may be re-enabled by specifying Z with no parameters.

The exchange fields used in the Z command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the old string and the new string.

A-49 V (UNCONDITIONAL CHARACTER REPLACEMENT, WITH LIST)

Used to insert, delete or replace characters on one or more lines starting from the first column of the edit window (see the SET WINDOW section for details on the W command). It is always used in combination with a second command which defines the range of lines over which the string exchange is to take place. The lines which are changed by this command are displayed to your terminal.

- -

```
V/newstring          -----------Insert new string starting at first
                                column of window.

Vxxx/                -----------Delete number of characters specified
                                by number of characters of xxx starting
                                at first column of window.

Vxxx/newstring       -----------Replace number of characters specified
                                by number of characters of xxx with new
                                string starting at first column of window.

range command        -----------Range of lines over which exchanges are to
                                take place.
```

- -

xxx
Can be any alphanumeric characters since only the number of characters in xxx matters. xxx is not used as a pattern for a search, but rather, to specify the number of characters to be replaced or deleted at the start of the window by the new character string. xxx can be a maximum of 148 alphanumeric characters.

newstring
This is the newstring which is to either replace the number of characters specified by xxx or to be inserted at the first column of the window.

A-49 V (UNCONDITIONAL CHARACTER REPLACEMENT CONTINUED)

range command
This is any command which will cause the pending line pointer to advance through the source file.

EXAMPLES:

FILEA contains the following three lines of text:

```
AAAAAAAAAAAAAAA
BBBBBBBBBBBBBBB
CCCCCCCCCCCCCCC
```

1) V/newstring example (insert new string).

```
/W5,150           -----------Set window between columns 5 to 150.
/P                -----------Display pending line.
 AAAAAAAAAAAAAAA --------Pending line displayed.
/V/XXX            -----------Insert XXX starting in the first column of
                             the window over the range specified in the
                             next command.
/+2               -----------Advance pending line down two lines.
 AAAAXXXAAAAAAAAAAA -----Changed line displayed.
 BBBBXXXBBBBBBBBBBB -----Changed line displayed.
 CCCCCCCCCCCCCCC    -----New pending line.
```

2) Vxxx/example (delete characters).

```
/W1,3             -----------Set window between columns 1 to 3.
/P                -----------Display pending line.
 AAAAAAAAAAAAAAA --------Pending line displayed.
/V12345/          -----------Delete first five characters starting in
                             the first column of the window over the
                             range specified in the next command.
/FEOF             -----------Find EOF.
 AAAAAAAAAA       -----------Changed line displayed (first 5
                             characters deleted).
 BBBBBBBBBB       -----------Changed line displayed (first 5
                             characters deleted).
 CCCCCCCCCC       -----------Changed line displayed (first 5
                             characters deleted).
EOF               -----------End of file.
```

3) Vxxx/newstring example (replace characters).

```
/W1,3             -----------Set window between columns 1 to 3.
/P                -----------Display pending line.
 AAAAAAAAAAAAAAA
/VAA/1111         -----------Replace first two characters starting in
                             the first column of the window over the
                             range specified in the next command with
                             the string 1111.
```

A-49 V UNCONDITIONAL CHARACTER REPLACEMENT CONTINUED)

```
/+              -----------Advance pending line one line.
  1111AAAAAAAAAAAAAA ------Changed line displayed.
  BBBBBBBBBBBBBBB   ------New pending line.
/W5,150         -----------Set window between columns 5 to 150.
/VBB/2222       -----------Replace first two characters starting in
                           the first column of the window over the
                           range specified in the next command with
                           the string 2222.
/+              -----------Advance pending line one line.
  BBBB2222BBBBBBBBB  -----Changed line displayed.
  CCCCCCCCCCCCCCC    -----New pending line.
/1               ----------Make line one pending line and display it.
  1111AAAAAAAAAAAAAA ------Pending line displayed.
/L5              ----------Display five lines of text.
  1111AAAAAAAAAAAAAA
  BBBB2222BBBBBBBBB
  CCCCCCCCCCCCCCC
EOF              ----------End of file.
```

COMMENTS:

The exchange field starts at the first column of the window, but is not limited by the width of the window (note example 2 above).

The default print device is the terminal that you input the commands at, but by using the delete lines command (D), the advance line commands (+ or /), or specifying a specific line number for the pending line pointer to move to (n), the optional LU parameter in these commands can be invoked to send the changed lines to the printer or a disk file.

The exchange fields used in the V command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the xxx parameter with the new string.

A-50 U (UNCONDITIONAL CHARACTER REPLACEMENT, WITH LIST)

Used to insert, delete or replace characters on one or more lines starting from the first column of the edit window (see the SET WINDOW section for details on the W command). It is always used in combination with a second command which defines the range of lines over which the string exchange is to take place. It is the same as the V command except that there is no listing of the changed lines.

- -

U/newstring	----------Insert new string starting at first column of window.
Uxxx/	----------Delete number of characters specified by number of characters of xxx starting at first column of window.
Uxxx/newstring	----------Replace number of characters specified by number of characters of xxx with new string starting at first column of window.
range command	----------Range of lines over which exchanges are to take place.

- -

xxx
Can be any alphanumeric characters since only the number of characters in xxx matters. xxx is not used as a pattern for a search, but rather, to specify the number of characters to be replaced or deleted at the start of the window by the new character string. xxx can be a maximum of 148 alphanumeric characters.

newstring
This is the newstring which is to either replace the number of characters specified by xxx or to be inserted at the first column of the window.

range command
This is any command which will cause the pending line pointer to advance through the source file.

- -

EXAMPLES:

FILEA contains the following three lines of text

```
AAAAAAAAAAAAAAA
BBBBBBBBBBBBBBB
CCCCCCCCCCCCCCC
```

A-50 U (UNCONDITIONAL CHARACTER REPLACEMENT, WITHOUT LIST) (Cont'd.)

1) U/newstring example (insert new string).

```
/W5,150               ----------Set window between columns 5 to 150.
/P                    ----------Display pending line.
  AAAAAAAAAAAAAAA        -------Pending line displayed.
/U/XXX                ----------Insert XXX starting in the first column
                                of the window over the range specified
                                in the next command.
/+2                   ----------Advance pending line down two lines.
  CCCCCCCCCCCCCCC         ------New pending line.
/1                    ----------Make line one pending line and display it.
  AAAAXXXAAAAAAAAAAA      ------Pending line displayed.
/L5                   ----------Display five lines of text.
  AAAAXXXAAAAAAAAAAA
  BBBBXXXBBBBBBBBBBB
  CCCCCCCCCCCCCCC
EOF                   ----------End of file.
```

2) Uxxx/example (delete characters).

```
/W1,3                 ----------Set window between columns 1 to 3.
/P                    ----------Display pending line.
  AAAAAAAAAAAAAAA         ------Pending line displayed.
/U12345/              ----------Delete first five characters starting in
                                the first column of the window over the
                                range specified in the next command.
/FEOF                 ----------Find EOF.
EOF                   ----------End of file.
/1                    ----------Make line one pending line and display it.
  AAAAAAAAAA               -----Pending line displayed.
/L5                   ----------Display five lines of text.
  AAAAAAAAAA               -----Changed line displayed (first 5
                                characters deleted).
  BBBBBBBBBB               -----Changed line displayed (first 5
                                characters deleted).
  CCCCCCCCCC               -----Changed line displayed (first 5
                                characters deleted).
EOF                    ---------End of file.
```

3) Uxxx/newstring example (replace characters).

```
/W1,3                 ----------Set window between columns 1 to 3.
/P                    ----------Display pending line.
  AAAAAAAAAAAAAAA        -------Pending line displayed.
/UAA/1111             ----------Replace first two characters starting
                                in the first column of the window over
                                the range specified in the next command
                                with the string 1111.
/+                    ----------Advance pending line one line and display it.
  BBBBBBBBBBBBBBB         ------New pending line (changed line not
                                displayed).
```

A-50 U (UNCONDITIONAL CHARACTER REPLACEMENT, WITHOUT LIST (Cont'd.)

```
/W5,150                -----------Set window between columns 5 to 150.
/UBB/2222              -----------Replace first two characters starting
                                  in the first column of the window over
                                  the range specified in the next command
                                  with the string 2222.
/+                     -----------Advance pending line one line and display
                                  it.
  CCCCCCCCCCCCCCC          -------New pending line (changed line not
                                  displayed).
/1                     -----------Make line one pending line and display it.
  1111AAAAAAAAAAAA         -------Pending line displayed.
/L5                    -----------Display five lines of text.
  1111AAAAAAAAAAAA         -------Changed line displayed (AA replaced with
                                  1111 starting in column 1).
  BBBB2222BBBBBBBB         -------Changed line displayed (BB replaced with
                                  2222 starting in column 5).
  CCCCCCCCCCCCCCC          -------Unchanged line displayed.
EOF                    -----------End of file.
```

COMMENTS:

The exchange field starts at the first column of the window, but is not limited by the width of the window (note example 2 above).

The exchange fields used in the U command are not tabbed, so that the tab character can be used like any other character in the exchange string.

Only the first occurrence of the current delimiter is recognized. It is assumed to separate the xxx parameter with the new string.

A-51 A (ABORT EDIT SESSION)

Used to abort the edit session and leave the original source file unchanged.

- -

A

- -

No parameters required

- -

A-51 A (ABORT EDIT SESSION CONTINUED)

EXAMPLE:

FILEA contains the following two lines of text:

```
THIS EXAMPLE DEMONSTRATES
THE ABORT COMMAND

:RU,EDITR            ------FMGR command to call EDITR.
SOURCE FILE?         ------EDITR requesting name of file to be edited.
/FILEA               ------File to be edited.
  THIS EXAMPLE DEMONSTRATESFirst line of file.
/-2                  ------Delete two lines of text.
EOF                  ------End of file mark indicates all text has been
                           deleted.
/A                   ------Abort edit session leaving original file
                           unchanged.
EDITR ABORTED        ------EDITR response verifying edit session has been
                           aborted.
:LI, FILEA           ------FMGR command to display contents of FILEA.

FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001  THIS EXAMPLE DEMONSTRATES
0002  THE ABORT COMMAND
```

A-52 EC (END EDIT AND CREATE A FILE MANAGER FILE)

Ends the edit session and creates a named File Manager file for storage of the contents of the destination work area.

- -

ECnamr

- -

namr
file name[:security code[:cartridge reference]]

(Refer to Section V for description of NAMR parameter).

- -

EXAMPLE:

```
:RU, EDITR          ------FMGR command to run EDITR.
SOURCE FILE?        ------EDITR request for name of file to be edited.
/0                  ------Put empty file into EDITR's source work area.
EOF                 ------End of file.
/ LINE ONE          ------Insert first line of text.
/ LINE TWO          ------Insert second line of text.
/EC&FILEA:AA:1000   ------End edit session and store destination work area
                          into created file named &FILEA with security code
                          AA on cartridge with reference number 1000.
```

A-52 EC (END EDIT AND CREAT A FILE MANAGER FILE CONTINUED)

```
END OF EDIT          ------EDITR response indicating edit session is
                           terminated.

:LI,&FILEA:AA:1000 ------FMGR command to list the contents of file named
                           &FILEA.

FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002
0001  LINE ONE
0002  LINE TWO
```

A-53 ER (END EDIT AND REPLACE OLD FILE WITH NEW FILE)

Ends an edit session and replaces the FMGR file with the edited version stored in the destination work area.

- -

```
ER           -------------End edit session and replace disc resident file
                          with edited version.

ERnamr       -------------End edit session and replace disc resident file
                          with edited version.
```

- -

namr
filename[:security code[:cartridge reference]]

(Refer to Appendix D for a description of NAMR parameter).

- -

EXAMPLE:

FILEA contains the following two lines of text:

```
LINE ONE
LINE TWO
```

1) ER example.

```
:RU,EDITR        ------FMGR command to run EDITR.
SOURCE FILE?     ------EDITR request for name of file to be edited.
/FILEA           ------Name of file to be edited.
  LINE ONE       ------First line of file.
/+               ------Advance pending line one line and display it.
  LINE TWO       ------Pending line displayed.
/ LINE THREE     ------End edit session and replace contents of disc
                       resident file with contents of destination work area.
```

A-53 ER (END EDIT AND REPLACE OLD FILE WITH NEW FILE CONTINUED)

```
END OF EDIT       ------EDITR response indicating edit session has terminated.

:LI,FILEA         ------FMGR command to list the contents of FILEA to terminal
                        CRT.

FILEA  T=00004 IS ON CR01000 USING 00001 BLKS R=0002

0001  LINE ONE
0002  LINE TWO
0003  LINE THREE
```

2) ERnamr example (Editing a file with a security code).

```
:RU,EDITR         ------FMGR command to run EDITR.
SOURCE FILE?      ------EDITR request for name of file to be edited.
/FILEA            ------Name of file to be edited.
  LINE ONE        ------First line of file.
/+                ------Advance pending line one and display it.
  LINE TWO        ------Pending line displayed.
/ LINE THREE      ------Insert new line of text.
/ERFILEA:SM       ------End edit session and replace disc file named
                        FILEA with edited version (note that security
                        code, SM, is required).
END OF EDIT       ------EDITR response indicating edit session has
                        terminated.
```

A-54 EDITR IN BATCH ENVIRONMENT

In batch mode, commands to EDITR are supplied as part of the job command file; that is, all EDITR commands must be supplied before the job is begun. Once the job is under way, you cannot interact with the computer to change the course of the processing.

All EDITR commands function the same in batch as they do in the interactive mode. You need not supply the EDITR prompt (/) on the input commands.

It is important that you know beforehand what is going to result from the edit commands in the job deck. Any condition which causes an error message causes EDITR to abort since there is no possibility of user intervention to fix the error. You must also insure that EDITR goes through the normal terminate sequence initiated by EL, EC, or ER. If an edit deflects this sequence, EDITR aborts.

If EDITR is run in batch mode without spooling, File Manager commands and EDITR commands must be read from an external device. Under spooling, however, EDITR commands can be read from a file if the file appears to be an external device. This is done by setting up an input spool to reference a file name as shown in the examples which follow. The default attributes should always be specified for the input spool.

A-54 EDITR IN BATCH ENVIRONMENT

1) EDITR WITHOUT SPOOLING

```
:RU, EDITR,5            ----- Card Reader defined as LU 5.

(Job deck in card reader)
```

2) EDITR WITH SPOOLED JOB

```
:JO,SMITH
:RU,EDITR,5             ----- The EDITR commands appear in the job
                              stream which may be a file or a card
 EDITR commands               deck, or tape, etc.

 ECnamr
:EOJ

:JO,SMITH
:LU,50,COMND            ----- Set input spool to reference file COMND
                              containing the EDITR commands.
:RU,EDITR,50            ----- In this example, EDITR acts as if it
                              were reading from LU 50, but it is
                              actually reading records from COMND
                              through the spool monitor driver.

:EOJ
```

A-55 EDITR IN A MULTI-TERMINAL ENVIRONMENT

In RTE-IVB, both the Session Monitor and the Multi-Terminal Monitor (MTM) allow any terminal user to operate independently of other users. Several users then can run the same program at the same time if each terminal has a separate copy of the program.

When the user types in the command to run EDITR,

```
:RU,EDITR
```

a program named EDIxx is created and scheduled to run at terminal xx, where xx is the LU of the terminal when operating under the Multi-Terminal Monitor or the session identification number when operating under the Session Monitor. When EDIxx is finished, the ID segment is automatically returned to the system.

A-56 EDITR IN A MULTIPOINT ENVIRONMENT

In a multipoint environment, several special considerations apply to EDITR operations. A terminal is defined to be in a multipoint environment if its I/O is being processed through driver DVR07. To determine if a terminal is operating under multipoint, observe the transmit break light on the terminal. If it is blinking intermittently, the terminal is probably a multipoint terminal.

When the EDITR is invoked from a multipoint terminal, several actions are performed for the user.

1. The intrinsic tab function of the terminal is enabled which allows the user to use the TAB key on the terminal and set and clear tab stops with the SET TAB and CLEAR TAB keys on the terminal (the EDITR's tab character (;) remains on).

2. The INSERT CHAR, DELETE CHAR, and CLEAR DSPLY keys on the terminal are enabled for use as editing functions.

3. The Q and O commands, which will be described later in this section, are enabled for character edits.

Functions which do not work in a multipoint environment include:

1. The CNTL and ESC keys do not work in a multipoint environment, therefore, the Q and O commands must be used for character edits.

2. The INSERT LINE and DELETE LINE keys do not work, therefore, the EDITR commands to insert and delete lines must be used.

3. The RETURN key (carriage return) is not used for data transmittal in a multipoint environment. The ENTER key is used to transmit text to the multipoint controller.

Furthermore, the characters that the EDITR will process are usually delimited by the left margin on the left and the current cursor position on the right.

Except for the difference cited above, all of the other EDITR commands may be used in a multipoint environment.

CHARACTER EDITS WITH THE Q AND O COMMANDS

The Q and O commands are used for character edits in a multipoint environment. Both commands are used to edit the pending line. The only difference betwen them is that the Q command edits the pending line, whereas, the O command duplicates the pending line then edits the duplicate. Although only the Q command is discussed throughout the rest of this section, the discussion applies to the O command as well.

CHARACTER EDITS WITH THE Q AND O COMMANDS (CONTINUED)

When the Q or O command is entered, the pending line is displayed, along with a delimiter (GS) to the left of the line. The delimiter is not part of the text string, but it must be preserved to assure proper operation. The delimiter is represented as a pound sign (#) throughout the rest of this section. The EDITR will position the cursor underneath the first character of the line.

To retain the ending line as it is, immediately hit the ENTER key.
For example:

```
/Q
 #ABCDE        ------Cursor displayed under first character in line.
                     Press the ENTER key and line is retained as is.
/P             ------Display pending line.
  ABCDE        ------Pending line displayed.
```

DELETE CHARACTERS FROM THE END OF A LINE

To truncate characters from the end of a line, position the cursor immediately after the last character to be retained. Strike the ENTER key to enter all the characters between the left margin and the current cursor position. The intrinsic terminal key CLEAR DSPLY can be used to delete characters at the end of the line from the screen. After using the CLEAR DSPLY key, the ENTER key is used to enter the edited line. For example:

```
/Q             -----Enter Q to make character edit.
 #ABCDE        -----Pending line is displayed. Cursor is moved under
                    the D using the right arrow key (→) to delete D
                    and E from line. CLEAR DSPLY key can then be struck
                    to clear D and E from screen.
 ABC           ---- Edited line is displayed.
```

INSERT CHARACTERS WITHIN A LINE

To add characters in the middle of a line, the INSERT CHAR key may be used. Press the INSERT CHAR key. The red light above the key should come on. Move the cursor to the position where the characters are to be added, and type in the new characters. Finally, position the cursor at the end of the line and hit the ENTER key. The insert light will go off and the edited line will remain as the pending line. For example:

```
/Q                -----Enter Q to make character edit.
 #ABCDE           -----Pending line is displayed. Cursor
                       is moved under the C using the right
                       arrow key (→). INSERT CHAR key is
                       depressed and XXX is typed in.
ABCXXXDE          -----Edited line is displayed.
```

DELETE CHARACTERS WITHIN A LINE

To delete one or more characters, position the cursor under the character to be deleted and press the DELETE CHAR key. The character will be deleted from the display and the rest of the line will be shifted left to fill in the gap. After all of the desired deletions have been made, move the cursor to the end of the line and press the ENTER key. Do not delete the delimiter at the beginning of the line. For example:

```
/Q          -----Enter Q to make character edit.
 #ABCDE     ---- Pending line is displayed.  Cursor is moved under
                 the B using the right arrow key (-->).  DELETE CHAR
                 key is depressed three times to delete the B, C and
                 D.  Move the cursor to the end of the line and press
                 ENTER.
 AE         -----Edited  line is displayed.
```

TAB CONTROL IN MULTIPOINT ENVIRONMENT

When the EDITR is invoked in a multipoint environment, the TAB key on the terminal is enabled (the EDITR tab character remains on). The tab stops are initially set to columns 7 and 21. These may be changed using the SET TAB and CLEAR TAB keys on the terminal. The TAB key may be used to position the cursor at any time. It is equivalent to moving the cursor using the terminal keys with arrows on them.

APPENDIX B

DISC DESCRIPTION AND UTILITIES

B-1 INTRODUCTION.

B-2 This section contains a description of the HP7906 Disc Drive and the HP7925 Disc Drive. Utilities to spare tracks and save/restore disc cartridges are also included in this section.

B-3 DISC DRIVE DATA.

B-4 Both the 7906 and 7925 disc drives read information off disc or write information on the disc on command from the HP13037 Disc Controller (controller). As many as eight disc drives can be connected to one controller. Controller/disc drive communications occur on two buses. One set of read/write data lines per disc drive is used to transfer read/write data.

B-5 The disc drive is composed of eight systems; a spindle rotating system, a head-positioning system, an input/output (I/O) control system, a read/write system, a sector sensing system, a power system, an air circulation system, and a fault detection system. The spindle rotating system rotates the discs at a speed of 3600 revolutions per minute. The head positioning system positions the heads in response to controller direction and, under emergency fault conditions, retracts the heads off the discs. The I/O control system interfaces between the tag and control buses and the internal disc drive systems. The read/write system reads and writes data from the read/write data lines to the discs or vice versa. The sector sensing system constantly monitors the identity of the sectors currently under the heads. This information is transmitted to the controller through the I/O control system on request. The read/write system also requires this information to know on which sector of the disc to read or write. The power system supplies power to the disc drive components. The air circulation system supplies cooling air to the circuits and filtered air to the discs. The fault detection system is composed of several subsystems which sense fault conditions and light indicators, retract the heads, or take other appropriate action when a fault occurs.

B-6 DISC ORGANIZATION.

B-7 The smallest addressable data storage area in the disc drive is a sector. Accessing a sector is accomplished by specifying the address of the cylinder, head, and sector.

B-8 Each sector contains a sector address field, a data field, and data checking and error correction fields. The sector address field contains the cylinder, head, and sector addresses of the sector, as well as indicators for spare, defective, and protected tracks. The data field stores 128 words of data. Each data word is defined as 16-bits. (See Figure B-1).

B-9 Each track on the 7906 disc drive is divided into 48 equal data sectors. Each track on the 7925 disc drive is divided into 64 equal data sectors.

B-10 CONTROLLER SELECTION OF A DISC DRIVE.

To enable the controller to communicate with one disc drive at a time, each disc drive is assigned an identity, from 0 through 7. The controller uses this identity to select the disc drive with which it wants to communicate. The 7925 disc drive is assigned the value 0 and the 7906 is assigned the value 1.

B-11 7906 DISC DESCRIPTION.

B-12 The 7906 disc drive contains two discs; one removable and the other fixed. It accesses the data on three physical (four logical) surfaces with three physical (four logical) read/write heads. Head-positioning and sector-counting information is derived from the fourth (servo) surface through the servo head on the fixed disc. There are 411 cylinders available for information storage. The cylinder address range is from zero to 410. Each cylinder consists of four tracks, one on each disc surface. Each track is divided into 48 sectors. Sectors are addressed by specifying a head and sector address within a cylinder. Head addresses range from zero to three and sector addresses range from zero to 47.

Figure B-2 shows the 7906 Disc surface logical unit designation.

B-13 7925 DISC DESCRIPTION.

B-14 The HP7925 Disc Drive is a single unit that contains five disc data platters and two platters for media protection only. Each data disc platter on the 7925 has two surfaces; however, one surface is used for timing purposes and is not available for data recording. Therefore, a single HP7925 Disc Drive contains 9 surfaces (9 heads), and 823 cylinders, giving 7,407 tracks. Note that a cylinder consists of one track from each surface. For example, cylinder #7 would be made up to the eighth track on surface 0, the eighth track on surface 1, the eighth track on surface 2, the eighth track on surface 3, and the eighth track on surfaces 4, 5, 6,7, and 8.

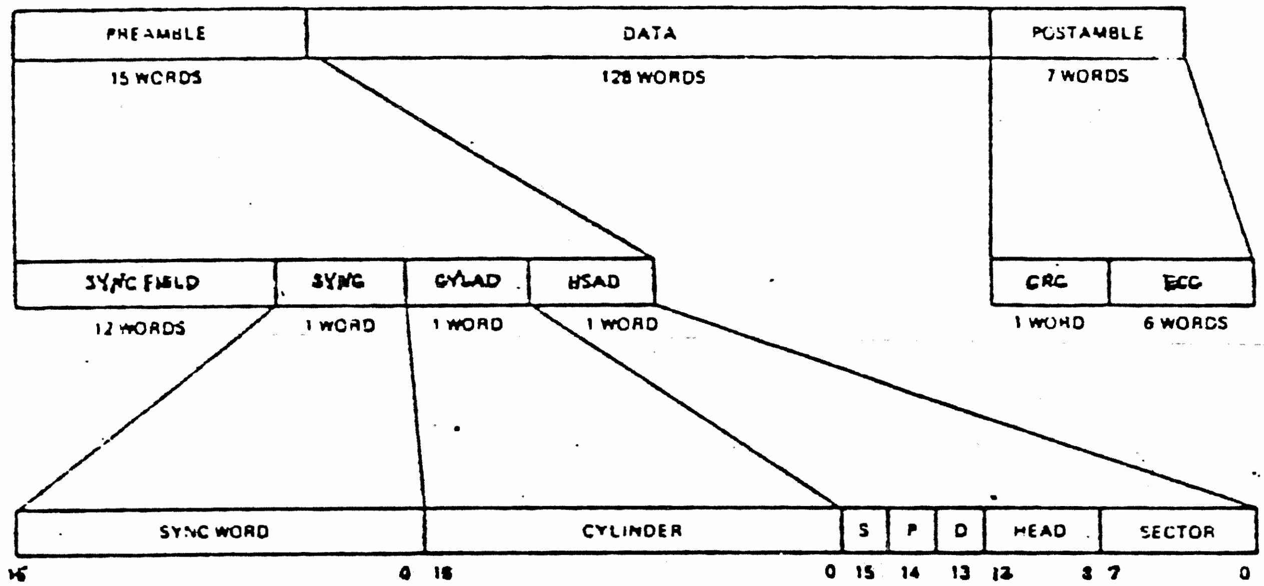
Figure B-3 shows the 7925 Disc structure.

Table B-3 gives the number of tracks per LU for the 7925 disc drive.

B-15 DISC DRIVE CAPACITIES.

B-16 There are 6144 words/track for the 7906 disc drive and 8192 words/track for the 7925 disc drive.

Refer to Table B-1 for the 7906 Disc Drive data capacity and Table B-2 for the 7925 Disc Drive data capacity.



PREAMBLE - 15 WORDS FOR SYNCHRONIZATION AND ADDRESSING

DATA - 128 WORDS OF DATA

POSTAMBLE - DATA CHECKING AND ERROR CORRECTION INFORMATION

SYNC FIELD - 12 WORDS (160 BITS) OF 0's

SYNC - SYNC WORD - 100376_8 IF ECC FIELD IS VALID

- 100377_8 OTHERWISE

CYLAD - CYLINDER - CYLINDER ADDRESS

HSAD - HEAD ADDRESS (LOGICAL)
SECTOR ADDRESS (LOGICAL)
S - IF "1" SPARE TRACK IN ACTIVE USE
P - IF "1" PROTECTED TRACK
D - IF "1", DEFECTIVE TRACK

CRC - CYCLIC REDUNDANCY CHECK - 1 WORD OF CHECK INFORMATION

ECC - ERROR CORRECTION CODE - 6 WORDS OF CHECK AND CORRECTION INFORMATION

Figure B-1. Sector Recording Format

Table B-1. 7906 Disc Drive Data Capacity

ITEM	TOTAL BITS PER	DATA BITS PER	DATA BYTES PER	DATA WORDS PER	DATA SECTORS PER	DATA TRACKS PER	DATA RECORDING SURFACES PER
BYTE	8	8					
WORD	16	16	2				
SECTOR	2368	2048	256	128			
TRACK	113664	98304	12288	6144	48		
SURFACE*	45465600	39321600	4915200	2457600	19200	400**	
CARTRIDGE	90931200	78643200	9830400	4915200	38400	800	2
DRIVE	18162400	157286400	19660800	9830400	76800	1600	4***

* Either surface of the removable disc or the lower surface of the fixed disc.

** Plus 11 spare tracks.

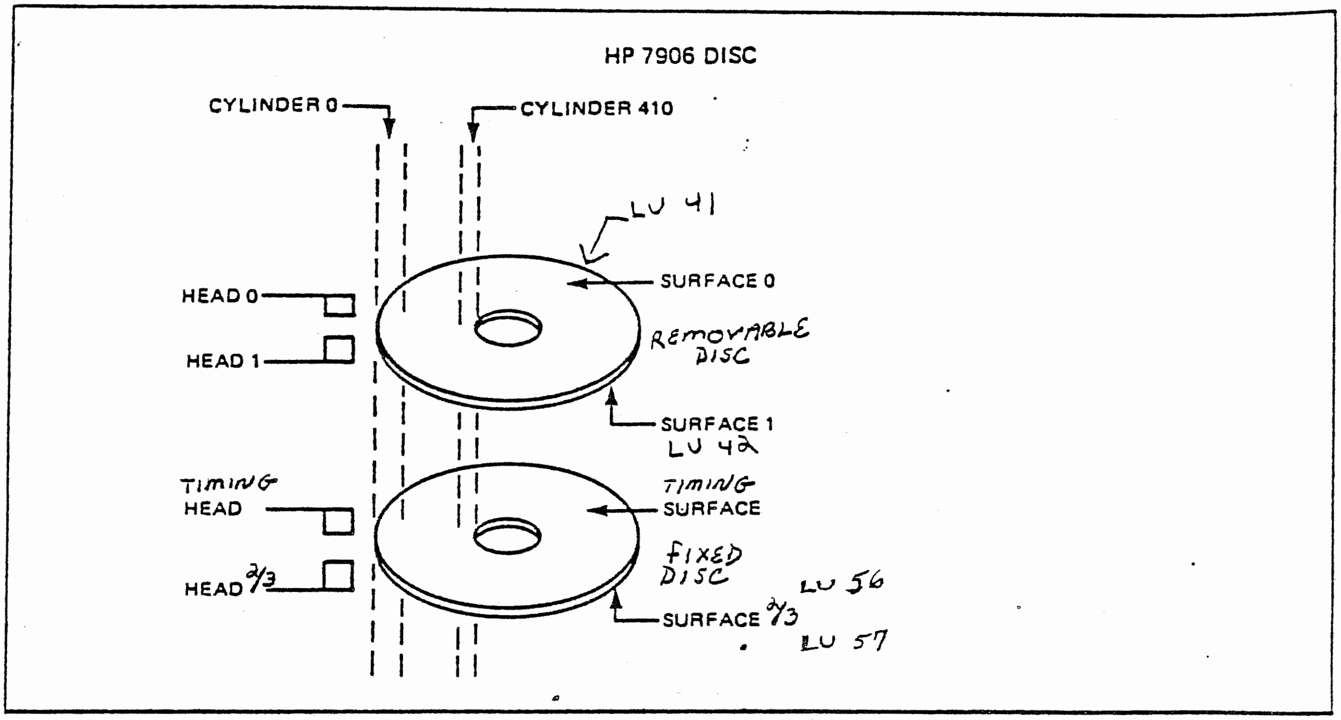
*** The upper surface of the fixed disc contains prerecorded head-positioning information. The lower surface is double density and therefore contains 800 tracks. The controller software, however, treats it as 2 logical surfaces. Therefore we have 4 recording surfaces logically and 3 physically.

Table B-2. 7925 Disc Drive Data Capacity

	DATA BITS PER	DATA BYTES PER	SECTORS PER	TRACKS PER
BYTE	8			
SECTOR	2,048	256		
TRACK	131,072	16,384	64	
SURFACE	106,713,680	13,352,960	52,160	815
DRIVE	960,423,120	120,176,640	469,440	7,335

* Total number of Tracks per surface is 823, 8 of which are utilized as spares or for defective track allocation. 815 tracks per surface (minimum) are guaranteed to be good.

Figure B-2. 7906 Disc Surface Logical Unit Designation



LU #	ALLOCATED SPARE (No. of Tracks)
41	400 Tracks + 11 Spares
42	400 Tracks + 11 Spares
56	400 Tracks + 11 Spares
57	400 Tracks + 11 Spares

Figure B-3. 7925 Disc Structure

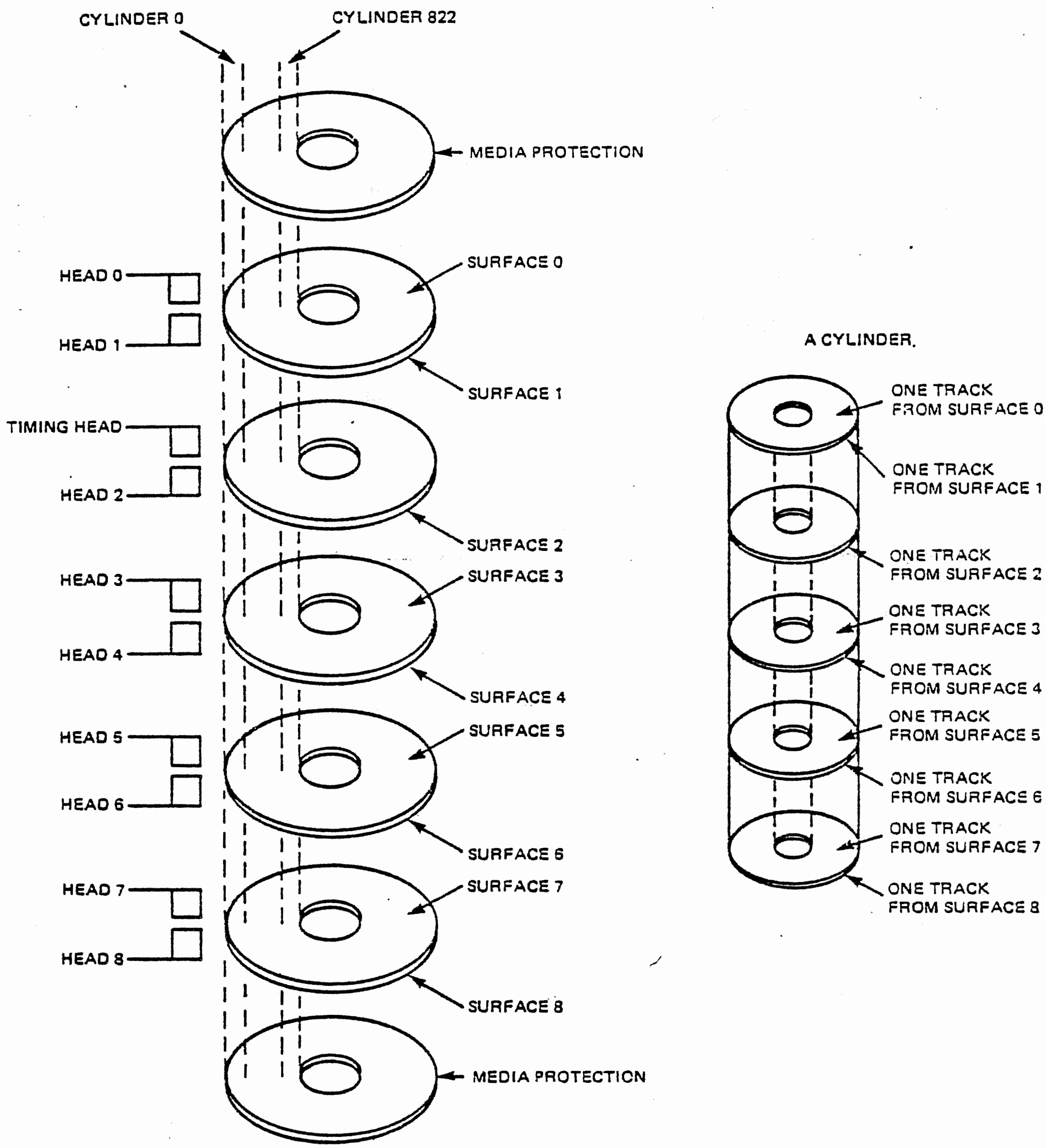


Table B-3. 7925 Disc Drive Allocated Space

LU #	ALLOCATED	SPACE (NO. OF TRACKS)	START CYLINDER
2	256 Tracks	+ 5 Spares	0
35	400 Tracks	+ 5 Spares	29
44	400 Tracks	+ 5 Spares	74
37	800 Tracks	+ 10 Spares	119
38	800 Tracks	+ 10 Spares	209
39	400 Tracks	+ 5 Spares	299
45	214 Tracks	+ 2 Spares	344
46	214 Tracks	+ 2 Spares	368
47	214 Tracks	+ 2 Spares	392
3	256 Tracks	+ 5 Spares	416
48	764 Tracks	+ 19 Spares	445
40	400 Tracks	+ 5 Spares	532
50	212 Tracks	+ 4 Spares	577
51	107 Tracks	+ 1 Spare	601
52	53 Tracks	+ 1 Spare	613
53	53 Tracks	+ 1 Spare	619
54	53 Tracks	+ 1 Spare	625
55	53 Tracks	+ 1 Spare	631
43	800 Tracks	+ 10 Spares	637
36	800 Tracks	+ 10 Spares	727

Note: All 7925 LU's use 9 surfaces.

B-17 TRACK SPARING.

Disc track sparing is often a misused and misunderstood term. Track sparing is the process of automatically seeking to a spare track upon detection of a defective track which cross references itself to the spare track. For the 7906/7925 family of disc drives, the process is transparent to the user because of an "intelligent" disc controller. The controller performs an automatic "seek to read" or "seek to write" to tracks which replace defective tracks.

If RTE encounters an unflagged defective track, it will report the transfer error with a "TR nnnn" message. However, if RTE encounters a defective track which was flagged defective, and a spare track had been activated, then sparing will occur.

There are no utility programs that operate under RTE which will flag defective tracks and activate the appropriate spares.

B-18 All new 7906 removable discs have to be formatted and initialized as well as tested for defective tracks. Non-HP discs (i.e., MEMOREX) require setting up the proper format in the preamble of the physical sectors.

Every user should make sure that his disc (assuming it is a new disc) is formatted and initialized BEFORE storing any information on the disc. A record of defective tracks should be kept on the underside of the disc.

A program called INITD will accomplish the above task. Any track that is detected as being bad is marked as defective and a spare track is activated and assigned to the defective track. This program allows up to 6 bad tracks on a surface. A disc with more than 6 bad tracks on a single surface should not be accepted from a vendor.

Each bad track and its associated spare will be printed out on the CRT by this program. Additionally, this program will set up the disc so that LU #41 will reside on the top surface of the removable disc and LU #42 will reside on the lower surface of the removable disc. A record of bad tracks and spares should be put on a label and appended to the bottom surface of the removable disc.

This program will function under control of RTE-4B.

B-19 REMOVABLE DISC FORMATTING AND INITIALIZATION.

The procedures for formatting and initializing a user's removable disc are as follows: (Note that this program requires exclusive use of the system).

a. Place the operating system in the File Manager mode. When system is in the File Manager mode the File Manager prompt character (: colon) will appear on the CRT.

b. Remove the current removable disc and mount the disc to be initialized and formatted.

c. When the DRIVE READY light appears, key in the following command:

```
TR,/INITD
```

The following dialogue will occur between the program and the user; (user response is is underlined).

GAC RTE 4 DISC INITILIZATION PROG.

TO BE USED TO INITIALIZE AND FORMAT A REMOV DISC FOR RTE 4

BAD TRKS WILL BE AUTOMATICALLY SPARED

IS DISC TO BE FORMATTED MOUNTED? (Y, N)

Y

OK TO WRITE ON REMOVABLE DISC? (Y, N)

Y

After the disc has been initialized and formatted, the program will print:

REMOVE DISC (2 SFC) FORMATTED FOR HP DISC DRIVE

DISC STILL TO BE INITIALIZED TO LU 41, 42, FOR FMGR

*** REMOVABLE DISC NOW CONFIGURED FOR RTE 4 ******

**** LU41 AND LU42 ALSO INSERTED AND INITIALIZED FOR USER

After this program has terminated, the user removable disc will have 400 tracks on LU 41 and 400 tracks on LU 42.

This disc is now available for reading and writing operations within the RTE Operating System.

B-20 The HP utility FORMT will perform track sparing for the 7925 disc. (It can also be used for the 7906 disc). Bad tracks are reported by RTE-IVB while the disc is in use. FORMT will substitute a spare track for the defective track and will copy as much data from the bad track as possible. Offset head reads are used in the recovery process and it is often possible to completely recover the data. In cases where this is not possible, usually only a single block will be lost.

B-21 To run FORMT, use the following:

:RU, FORMT [,log lu] [,SP,disc lu, track]]

After invoking the SP command, FORMT checks for a valid disc LU. If you attempt to spare a floppy disc or if the disc LU is not in your SST (under Session Monitor), the following message is printed.

INVALID DISC LU

You must then specify a valid disc LU.

After inputting an invalid parameter and/or in the interactive mode, FORMT asks:

```
TRACK TO BE SPARED?
```

Entering ?? to the query returns to you the track range for the LU. Then enter the track (logical track) found defective on the disc LU. If the track number of a good track is entered, it will be spared and cannot be recovered as a good track. If the track does not lie within the bounds of the disc LU specified, the messages are displayed:

```
INVALID TRACK #

ENTER BAD TRACK # X -- XXXX
```

You can then enter a bad track within the range specified.

CAUTION

> Once a track is spared, it is marked defective permanently. This process is irreversible. Do not attempt to spare the same track twice.

The program copies data block by block from the bad track to spared track. If all the information on the tack cannot be recovered, the following warning message is issued and FORMT continues:

```
WARNING ALL INFORMATION ON TRACK NOT SUCCESSFULLY RECOVERED
```

When the program completes, the sparing is reported:

```
BAD TRACKS SUBCHANNEL XX

LU XX          LOGICAL       CYL        HEAD        UNIT/ADDR

BAD TRACK       XXXX         XXXX        XX             XX

SPARED TO       XXXX         XXXX        XX             XX
```

If in the interactive TASK? mode, the program asks for another FORMT task. Otherwise, FORMT terminates.

CAUTION

> All discs sharing the EQT of the specified disc LU are locked for the duration of the "SP" command. All loads, swaps, and other accesses to any of the discs on the EQT will not be allowed for the duration of the "SP" command.

B-22 Refer to the Hewlett Packard RTE-IVB Utility Programs Reference Manual (Part No. 92068-90010) for more details and error messages of FORMT.

B-23 SAVE/RESTORE HP PROCEDURES FOR 7906/7925 DISC.

B-24 The following Hewlett Packard procedures are used to save or restore disc cartridges on either the 7906 or the 7925 disc drive. These procedures are outlined in this appendix, however, further details may be obtained in the "RTE-IVB Utility Programs Reference Manual" (Part No. 92068-90010), the "READR/SAVER Utility Reference Manual" (Part No. 92068-90016), and the "RTE-IVB Terminal User's Reference Manual (Part No. 92068-90002).

B-25 LSAVE

B-26 LSAVE is used to save one disc LU or subchannel onto magnetic tape. The data is transferred starting at the magnetic tape's current position and the tape is not rewound after the operation. For each save you have the option of verifying that the data is correctly saved on tape.

The RUN command format for LSAVE follows. If optional parameters are omitted, commas must be used as place holders.

```
:RU,LSAVE [[,log lu],disc lu [,mt lu] [,VE] [,title]]
```

[log lu]	LU of the device where messages are sent. Optional parameters, default is the session LU of your terminal.
disc lu	Positive LU of the disc subchannel to be saved. Under Session Monitor, the disc lu must be in your SST (Session Switch Table).
[mt lu]	LU of the magnetic tape drive. Optional parameter, default = 8.
[VE]	To verify the tape file after a save. VE to verify. NO for no verify. Optional parameter, default = NO.
[title]	Up to 40 characters of label information stored in tape header, to identify the tape file. The utility also palces the date and run string preceding the title in the header for additional identification. Optional parameter, no default.

To have the program prompt you for the parameters, just enter "RU,LSAVE". Enter a valid response to each question or a space and RETURN to take the default value. If a response is not valid, the question will be repeated. If you enter at least one comma after "RU,LSAVE" but have an incomplete or invalid run string, the program will default all optional parameters and prompt you for any missing or invalid parameter. To stop the program enter /E,EN, or EX in repsonse to a program question.

When the program has completed, it will print the number of tracks saved on the logical device.

(Refer to the "RTE-IVB Utility Programs Reference Manual" Part No. 92068-90010).

The operating system is saved with the LSAVE utility. See Appendix C for further details.

B-27 USAVE.

B-28 USAVE saves all of the subchannels associated with a disc unit (disc drive). The save is done according to the track map table currently defined in the system. As in LSAVE, the data is saved starting at the current tape position and the tape is not rewound after the save. Each save has the option of being verified.

The RUN command format for USAVE follows. If optional parameters are omitted, commas must be used as place holders.

:RU,USAVE [[,log lu] ,disc lu[,mt lu][,VE][,title]]

[log lu] LU of the device where messages are sent. Optional parameter, default is the session LU of your terminal.

disc lu LU of any disc subchannel on the unit to be saved. USAVE will search the track map table and save all subchannels on the same unit as the LU given.

[mt lu] LU of the magnetic tape drive. Optional parameter, default = 8.

[VE] Verify the tape file after the save.
Enter VE to verify.
Enter NO for no verify.
Optional parameter, default = NO.

[title] Up to 40 characters of information are stored in the tape header to identify the tape file. The utility appends this information to the date and the run string in the tape header for identification.

To have the program prompt you for the parameters just enter "RU,USAVE". Enter a valid response to each question or a space and RETURN to take the default value. If the response is not valid, the question will be repeated.

If you enter at least one comma after "RU,USAVE", but have an incomplete or invalid run string, the program will default all of the optional parameters and prompt you for any missing or invalid parameters. To stop the program enter /E,EN, or EX in response to any program question.

When the program has completed, it will print the number of tracks saved from each lu of the disc unit.

(Refer to the "RTE-IVB Utility Programs Reference Manual" Part No. 92068-90010).

B-29 RESTR.

B-30 RESTR returns data saved on magnetic tape by USAVE or LSAVE to disc. However, RTE-IVB will not permit you to restore the system disc (LU2 or LU3). If LU2 or LU3 are included on a USAVE file, RESTR will skip LU 2 or LU3 and continue restoring the remaining subchannels.

Track sparing and initialization will not be performed.

The RUN command format for RESTR follows. If optional parametrs are omitted, commas must be used as placeholders.

```
:RU,RESTR [[,log lu] ,disc lu [,mt lu][,DE]]
```

[log lu] LU of the device where messages are sent. Optional parameter, default is the session LU of your terminal.

disc lu Destination disc LU of the restore. If the file being restored was saved with LSAVE,RESTR will restore the tape file to this disc LU. If the tape file was saved with USAVE, the track map table of the current system must match the track map table of the USAVE system as recorded in the tape header.

NOTE: In restoring either a USAVE tape file or a LSAVE tape file, the disc being restored must have the same track size (words per track) as the source save disc. Use READT/WRITT when the source and destination track size is different.

[mt lu] LU of the magnetic tape drive. Optional parameter default = 8.

[DE] Do not confirm the magnetic tape header. The program will list the tape file header and restore the disc LU of or unit without further operator intervention.

If the DE option is not specified, the program lists the tape file header and asks for confirmation.

OK?

The response may be:

YE[S]	-	Restore the magnetic tape file to the disc.
NO	-	Move the tape forward to the next file, print its header, and ask to continue.
EN,/E, or EX	-	Stop the program.

If more than "RU,RESTR" is entered in the run string but there are missing parameters, RESTR will ask for them. Enter a valid response or a space and RETURN to take the default value. If a response or parameter is not valid, the question will be asked again.

Enter /E,EN, or EX to any question to stop the program.

If the source LU on the tape file has more or less tracks than the destination lu, the program prints the decimal track sizes to your terminal:

```
XXXX TRACKS IN SOURCE LU
YYYY TRACKS IN DEST. LU
OK TO PROCEED?
```

If you enter "YES", the program copies tracks until it has restored the last track of the destination LU or the last track of the source file LU, whichever LU is smaller.

After any RESTR is complete, the program prints:

```
NO. OF TRACKS RESTORED XXXX
RESTR:  STOP 0077
```

CAUTION

> When restoring a disc LU with FMGR files and directory track(s), be careful if you try to restore to a disc LU with less tracks than the source LU. The directory track may not be copied and the restore would not be valid. However, to restore to a disc LU with more tracks, you must do a complete FMGR MC command to recover the valid directory. In the MC command, specify the correct number of tracks restored from the source disc LU.

(Refer to the "RTE-IVB Utility Programs Reference Manual" Part No. 92068-90010).

B-31 LCOPY.

B-32 LCOPY can copy one disc LU to another disc LU. The source and destination may or may not be suchannels at the same disc unit but the source and destination track size (words per track) must be the same.

Track sparing and initialization are not performed on-line.

The RUN command format for LCOPY follows:

```
:RU,LCOPY,source disc LU, destination disc LU
```

Source Disc LU	LU of the disc subchannel to copy from
Destination Disc LU	LU of the disc subchannel to copy to (cannot be LU2 or LU3).

All parameters are required in the run string. There are no defaults. Messages are always output to your terminal. LU2 and LU3 may be source disc LU's but never destination LU's.

If the source LU has more or fewer tracks than the destination LU, the program prints the decimal track sizes to your terminal.

```
XXXX TRACKS IN DEST LU
YYYY TRACKS IN SOURCE LU
OK TO PROCEED?
```

If you enter YES, the program copies tracks until it has copied the last track of the source LU or has copied to the last track of the destination LU, whichever LU is smaller.

After the LCOPY is complete the program prints:

```
XXXX TRACKS COPIED
     LCOPY:  STOP 0077
```

CAUTION

When copying a disc LU with FMGR files and directory track(s), be careful if you try to copy to a destination disc LU with less tracks than source LU. The directory track may not be copied and the copy would not be valid. However, to copy a destination disc LU with more tracks, you must do a complete FMGR MC command to recover the valid directory. In the MC command, specify the correct number of tracks copied from the source disc LU.

(Refer to the "RTE-IVB Utility Programs Reference Manual" Part No. 92068-90010).

B-33 WRITT.

B-34 WRITT saves FMP cartridges on magnetic tape.

The format for running WRITT is:

```
RU,WRITT [,DISC[,MT[,IH[,DC]]]]
```

where:

DISC — Is the disc cartridge to be saved. This parameter can be either a negative LU number or a positive CRN number.

WRITT will save the first non-session cartridge in the system cartridge list. If none is mounted, WRITT returns an error stating that a system disc cannot be saved.

MT — Is the logical unit number of the magnetic tape unit. You can specify either a "+" or a "-" LU number. Default is LU 8.

IH — Inhibits tape rewind before and after restoring a cartridge. Default is to rewind tape before and after cartridge restoration. This parameter can be used with the CN command to advance tape, e.g., CN,8,FF to advance tape one file. Tape rewind must be done separately with the "CN,8,RW" command.

DC — Is used to disable overlay check. If not specified, WRITT will check for a different FMP disc cartridge and, if there is one on tape, will wait for an answer. Enter yes to overwrite the tape and no to terminate WRITT. DC is used for saving several cartridges in a batch mode.

When a cartridge is saved on tape, WRITT creates a header at the start of the tape containing the CRN, the cartridge label, the date and time of day the cartridge was saved, and the cartridge type. For example:

```
CR 289 MANUF SAVED 10:45 AM FRI., 19 JAN., 1980 PR  --+
```

```
        +--Cartridge Lable        Cartridge Type  --+
     +--Cartridge Reference Number.
```

WRITT always rewinds the tape before and after a cartridge is saved. If the end of tape is reached before the entire cartridge is saved, WRITT sends a message instructing the user to mount another tape and enter GO after mounting the tape. WRITT will then continue saving the remainder of the disc cartridge.

If additional cartridges are to be saved on the same tape, use the IH parameter to inhibit tape rewind. Without the IH parameter, the magnetic tape is rewound before and after the disc save operation. You may want to advance the tape to a particular file before saving the current disc cartridge. Use the CN command to position tape at the desired location (e.g., :CN,8,FF will advance tape one file). After the cartridges have been saved, a file manager command (e.g., CN,8,RW) must be entered to rewind the tape.

CAUTION

> WRITT should not be used to update a cartridge at the beginning or in the middle of a multiple cartridge tape. This will corrupt the data located behind the updated cartridge. Saving multiple cartridges should be done at the end of the existing data.

(Refer to the "RTE-IVB Terminal User's Reference Manual" Part No. 92068-90002).

B-35 READT.

B-36 Utility READT restores disc cartridges from magnetic tape.

The format for running READT is:

```
RU,READT [,DISC[,MT[,type[,size[,IH]]]]
```

where:

DISC Is the disc cartridge where the saved FMP cartridge is to be restored. This parameter can be either a negative LU number or a positive CRN number.

This parameter must be a negative LU number. The CRN number is not allowed.

MT Is the logical unit number of the magnetic tape unit. You can specify either a "+" or "-" LU number. Default is LU 8.

TYPE Is an optional session environment parameter specifying the type of cartridge to be restored. Enter P for a private cartridge or G for a group cartridge. Default is to use the type specified in the tape header. If the size parameter is used, the position of this parameter must be maintained.

SIZE Is the desired size of the disc to which the magnetic tape contents are to be restored. The size is specified in number of tracks. Default is the size of the disc saved on tape.

IH Is used to inhibit tape rewind before and after restoring a cartridge. Default is to rewind tape before and after the disc cartridge restoration. This parameter can be used with the CN command to advance tape to the desired position, e.g., CN,8,FF to advance tape one file. Tape rewind must be done separately with the "CN,8,RW" commmand.

If it is necessary to overlay a currently mounted cartridge, consider the following items:

1. The last track of the cartridge to be overlaid cannot be moved.

2. The cartridge type (P or G) of the cartridge to be overwritten cannot be changed.

After the cartridge has been restored from tape to disc, the header created by WRITT and the LU to which the cartridge has been restored is displayed on the terminal.

If there is no disc cartridge large enough for the cartridge to be restored and if by moving the file directory the saved cartridge can be restored to an available cartridge, the following message will be displayed:

```
CRN ###### WAS SAVED FROM A XXXXX TRACK DISC
LAST TRACK USED IS          YYYYY
WOULD LIKE TO RESTORE A     ZZZZZ TRACK DISC
IS IT OKAY TO MOVE DIRECTORY TRACKS (YES OR NO)?
```

A YES answer will allow READT to restore the saved cartridge to the available disc. A NO answer will terminate READT.

The size parameter specifies the desired number of tracks required to restore the cartridge to. For example, a request is made for 203 track LU, READT will obtain a 203 track disc LU and place the first directory track at track 202.

In the event that a restore is made to a disc LU that has a sector/track value different from the sector/track value on magnetic tape, READT issues a message indicating that reformatting of the directory is necessary to maintain the integrity of the file structure:

```
TRACKS REFORMATTED FROM XXX SEC/TRK TO YYY SEC/TRK
```

READT will then proceed to restore the cartridge.

To restore FMP tracks on the system and auxilary disc (LU's 2 and 3), the user must be the system manager. The desired LU must be free of any activity, files may not be opened nor may there be ID segments pointing to files in the FMP track area. The first activity will cause a READ 009 error. The second case will cause READT to issue a message informing the user to "OF" the ID segments pointing to the FMP tracks. READT will not allow changing the starting location of FMP tracks or restoring a cartridge of a different sector/track value. Doing so will produce an error.

(Refer to the "RTE-IVB Terminal User's Reference Manual" Part No. 92068-90002).

B-37 READR/SAVER OVERVIEW.

B-38 The READR/SAVER package is a pair of file backup and restore utilities. The SAVER program saves disc files onto magnetic tape or minicartridge in a packed format. The READR program reads files from magnetic tape or minicartridge and places them on disc.

Another main feature of the READR/SAVER package is its flexibility in saving and restoring files selectively.

SAVER creates a file-name table of the files to be saved. It writes the file-name table on tape as a directory. SAVER then opens each file in turn, writes it to the tape, and verifies the file by comparing the tape data bit to the disc data. Optionally, the directory is the first file on tape and is in File Manager format. It can be accessed with a FMGR ST or DU command.

READR allows you to decide which files on tape are to be placed where on disc. An option to update your disc files is also included. If a file already exists on disc, and the update option is selected, the disc file will be replaced by the file on tape. The tape file will be verified before actual replacement takes place.

(Refer to the "READR/SAVER Utility Reference Manual", Part No. 92068-90016).

B-39 COMPATIBLE UPS UTILITIES.

B-40 The following GAC utilities are compatible with UPS Models AT230-1 and AT230-3 utilities GSAVE, GRSTR and DISCD. These utilities are MSAVE, MRSTR and MDISC, respectively.

These programs will SAVE/RESTORE using the same disc format as GSAVE, GRSTR and MDISC. However, they will also allow multiple users to operate concurrently.

B-41 MSAVE.

B-42 This utility is used for removable disc backup and is compatible with GAC utility GSAVE. MSAVE will copy the contents of a removable disc (logical units 41 and 42 or an operating system) on to a magnetic tape in the same format as GSAVE. The procedures to be used are as follows:

a. Mount a magnetic tape on the tape transport with "write ring" inserted.

b. Enter the following run command at the terminal:

```
RU,MSAVE
```

c. The program will display:

```
********************************************
**                                        **
**                                        **
**  GAC PROGRAM TO SAVE REMOVABLE DISC    **
**                                        **
**  MULTI TERMINAL PROGRAMMING STATION    **
**             VERSION                    **
**                                        **
**                                        **
********************************************


ENTER LU OF MAG TAPE DRIVE (8 OR 34):  LU #
            (default is LU8).

ENTER FILE ID
```

d. Enter up to 64 characters of information. This ID will be associated with LU 41 and LU 42 and will be written as a header in file 1 and file 2 of the magnetic tape.

e. The program will begin to copy LU 41 (top surface of the disc) to file 1 on the magnetic tape. After LU 41 has been copied, the program will begin to copy LU 42 (bottom surface of disc) to file 2 of the magnetic tape. After LU 42 has been copied, the magnetic tape will rewind.

As soon as it reaches the load point it will begin to verify the data written on the tape to the data stored on the disc. Logical unit 41 is verified (against file 1) then logical unit 42 is verified (against file 2). After the verification of the second file has terminated, the magnetic tape will rewind and the save program will be terminated.

B-43 MRSTR.

B-44 This utility is used for removable disc restoration from magnetic tape and is compatible with GAC utility GRSTR. MRSTR will restore discs that were saved via MSAVE. The procedures to be used are as follows:

a. Mount the proper magnetic tape (without "write ring") and ready the Tape Drive.

b. Mount the disc to the restored and wait until the DRIVE READY light on the Disc Drive is lit.

c. Enter the following run command at the terminal:

```
RU,MRSTR
```

d. The program will display:

```
************************************************
**                                            **
**                                            **
**  GAC PROGRAM TO RESTORE REMOVABLE DISC     **
**                                            **
**  MULTI TERMINAL PROGRAMMING STATION        **
**              VERSION                       **
**                                            **
**                                            **
************************************************


ENTER LU OF MAG TAPE DRIVE (8 OR 34):  LU#
            (default is LU8).


FILE #1 ID IS:

file ID for file 1 is displayed

OK TO CONTINUE? (Y OR N):
user enters "Y" to continue
```

At this point file 1 information will be transferred to the top surface (LU 41) of the removable disc.

Next, file 2 (LU 42) will be copied from magnetic tape to the bottom surface of the removable disc.

After LU 42 is restored, the tape will rewind and begin a verify operation:

File #1 will be verified with LU #41
File #2 will be verified with LU #42

After the verification operation has finished, the program will be terminated.

B-45 MDISC.

B-46 This utility is used to copy a removable disc to another removable disc and is compatible with GAC utility DISCD. MDISC copies the contents of a removable disc (logical units 41 and 42) to the 7906 fixed platter. By virtue of this procedure, the contents of the 7906 fixed platter are destroyed. MDISC then proceeds by copying the 7906 fixed platter to another removable disc. The procedures for MDISC are as follows:

a. Mount the removable disc to be copied (original) and ready the 7906 disc drive.

b. Enter the following at the terminal:

   :RU,MDISC

c. The program will display:

```
************************************************
**                                            **
**                                            **
**  GAC REMOVABLE DISC COPY-VERIFY PROGRAM    **
**                                            **
**  MULTI TERMINAL PROGRAMMING STATION        **
**               VERSION                      **
**                                            **
**                                            **
************************************************
```

7906 FIXED PLATTER WILL BE OVERWRITTEN

IS IT OK TO OVERWRITE? (Y OR N) : Y

IS DISC TO BE COPIED MOUNTED? (Y OR N) : Y
If "N" is entered the following message is printed:

PLEASE MOUNT PROPER REMOVABLE DISC

After the removable disc (MASTER) has been copied to the lower disc, the following messages will appear:

REMOVABLE DISC NOW BEING VERIFIED

VERIFY SUCCESSFULLY COMPLETED

d. Take out removable disc and mount disc to be written.

IS DISC TO BE WRITTEN MOUNTED? (Y OR N) : Y

At this point the user should have inserted the removable disc which is to become a copy of the master. After "Y" has been entered, the copy will begin.

When the copy has completed, the program will display:

COPY OF REMOVABLE DISC COMPLETED

REMOVABLE DISC NOW BEING VERIFIED

VERIFY SUCCESSFULLY COMPLETED

e. DO YOU WANT TO COPY TO ANOTHER REMOV DISC (Y OR N):

If "Y" is entered, the messages from Step d will be repeated for the next removable disc.

If "N" is entered, the program will terminate.

f. <u>ERROR PROCEDURE</u>

If there is an error, the following sequence of messages will be displayed:

VERIFY ERROR IN COMPARISON OF 2 DISCS

WORD NO ZZZZ OF TRACK TTTT IN ERROR

DO YOU WANT TO CONTINUE? (Y OR N): <u>Y</u> or <u>N</u>

If "Y" is entered, the program will continue with the verification beginning with the word following the failure.

If "N" is entered, the program will display:

VERIFICATION ABNORMALLY TERMINATED

Control will return to Step e.

APPENDIX C

OPERATIONAL PROCEDURES

C-1 INTRODUCTION.

C-2 This section describes the procedures necessary to allow the user to place the test system and operating system in the initialized state. The procedures described in this section are as follows:

a. System Power-Up

b. Computer System Boot-Up

c. System Power Down

d. Restoration of Operating System to Disc.

C-3 COMPUTER CONTROL AND INDICATORS.

C-4 Figure C-1 illustrates and describes the HP21MXE Computer front panel controls and indicators. It is not necessary to completely understand the function of all the controls and indicators to use the RTE. This section will only discuss those switches needed for power on, boot-up, and power off.

C-5 SYSTEM POWER-UP.

C-6 For a cold start operation, the following steps must be performed, in the order given, to correctly power up the system.

a. On the Power Control Panel, put 115 VAC POWER switch to ON position. At this time, the DOOR UNLOCKED indicator on the Disc Drive will light.

b. Insert the proper removable disc into the 7906 Disc Drive and place the Disc Drive RUN/STOP switch in the RUN position.

c. About 40 seconds after the DOOR UNLOCKED light goes out, the DRIVE READY light will come on. When the DRIVE READY indicator is lit, the operator should check to see that the power has been applied to all the instrumentation.

d. To operate the 7925 disc drive proceed as follows:

- o Set the RUN/STOP switch to STOP. Set the power switch to ON.
- o Install the 7925 disc pack as per instructions in the "7925 Disc Drive User's Manual" (Part No. 07925-90911).
- o Set the RUN/STOP switch to RUN. The DOOR UNLOCKED indicator will extinguish and after a 35 second start-up sequence is complete, the DRIVE READY indicator will light.

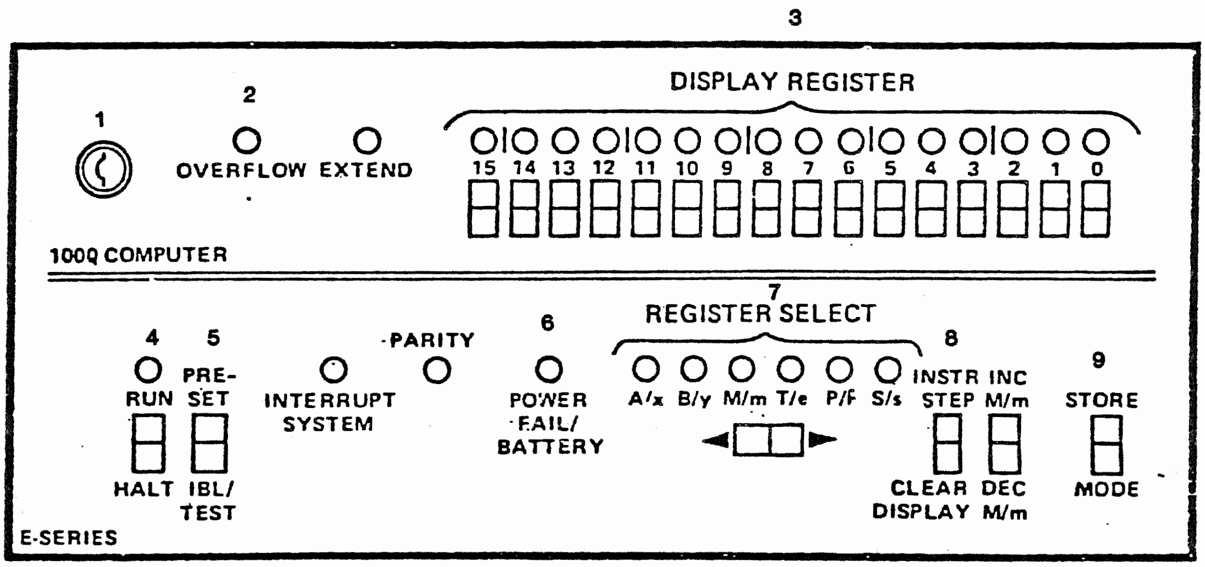


Figure C-1

HP21MXE Computer Front Panel, Controls and Indicators

C-7 COMPUTER INDICATORS - DESCRIPTION

Refer to Figure C-1.

1. KEY

Used to get to POWER and LOCK/OPERATE switches behind front panel. Power ON/OFF switch applies/removes AC line power to computer power supply.

OPERATE. Normal position when running computer. All front panel switches are enabled.

LOCK is same as OPERATE except RUN/HALT switch is disabled. Use to prevent someone halting system.

2. OVERFLOW

Lights if arithmetic overflow occurs. Usually can be ignored except during loading when it indicates an error in the initial binary loader, IBL.

3. DISPLAY REGISTER

Shows content of S register when computer is running, of selected register when halted. Press upper half of switch to set a bit to 1, lower half to set bit to 0. Light turns on when bit is set to 1.

4. RUN/HALT

Press RUN to begin programmed operation, press HALT to stop.

5. PRESET IBL/TEST

Press PRESET to initialize computer and clear parity and overflow bits while halted. Press IBL/TEST to write contents of selected loader ROM (read only memory) into memory, and perform automatic execution of firmware self-tests.

6. POWER FAIL

Lights when power restored after power off. Press HALT then PRESET to turn light off.

7. REGISTER SELECT

Selects register to display on 3. Press right half of switch to move listed selection to right; left half to move it to left. Light wraps around at either end of row.

8. INSTR STEP/CLEAR DISPLAY

Press INSTR/STEP to execute current instruction and advance P register to next instruction. If T lights when not selected, there is infinite indirect addressing and P register is not advanced. CLEAR DISPLAY clears each bit in display register 3 to zero.

9. STORE/MODE

You must press STORE to move contents of display register 3 to register selected by 7. If T selected, contents are stored in memory at address in M register. M is incremented by 1, display is unchanged. Press MODE to determine upper case or lower case indicators are to be used.

C-8 COMPUTER SYSTEM BOOT-UP.

C-9 Refer to Figure C-1 and proceed as follows:

a. Depress the HALT push button on the computer.

b. Move display indicator switch to the right so that display indicator S/s is lit.

c. Press INST STEP - CLEAR DISPLAY switch to CLEAR DISPLAY position to clear the 16 display registers.

d. Set display register switches 15, 12, 9, 6 to illuminate the respective display register indicators.

e. Press the STORE - MODE switch to the STORE position to store the contents of the display register into the S register.

f. Press the PRESET - IBL/TEST switch to the PRESET position, then to the IBL/TEST position, then back to the PRESET position, which will cause the disc ROM loader to be loaded.

g. Set the RUN - HALT switch to the RUN position. This causes the loaded disc ROM to be executed. The resident portion of RTE will be loaded and the following message will be displayed on the system CRT:

UNIVERSAL PROGRAMMING STATION
RTE 4B OP SYSTEM
AT230-5B1-1

GAC PART NO. A31U30121-1

GET SYSTEM PROMPT ON CRT
KEY IN TM YEAR, JULIAN DAY, HR, MIN, SEC

C-10 INSERT DATE/TIME.

The user keys in date and time according to the above format. After date-time has been entered and stored, the following will appear:

```
GTIME:  STOP 0000
::
```

The 2 other CRT terminals will display the following when the carriage return is depressed:

```
---REMOTE TERMINAL IS ON-LINE---
```

C-11 RTE 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 can also be performed from any of the multiple terminals. The entire set of supported languages is available in the multi-terminal environment.

It is important to note that some programs on the system are saved as Type 6 files as an alternative to being permanently loaded. The advantages to this are that ID segments are freed up in the system and valuable system track space is not tied up when the programs are not in use.

C-12 SYSTEM POWER-DOWN.

a. Place RUN - HALT switch on Computer to Halt position.

b. Place the 7906 Disc Drive RUN/STOP switch in the STOP position. Wait until DOOR UNLOCK indicator is lit.

c. Place the 7925 RUN/STOP switch to STOP. The DRIVE READY indicator will extinguish immediately. Allow the spindle to stop rotating (approximately 30 seconds). The DOOR UNLOCKED indicator will light indicating that the spindle has stopped. If desired, the disc pack may be left in the disc drive or may be removed. Remove the 7925 disc pack as per instructions in the "7925 Disc Drive User's Manual" (Part No. 07925-90911). Set the Power switch to OFF.

d. On the Power Control Panel, place 115 VAC POWER SWITCH to the OFF position.

C-13 SAVING AN OPERATING SYSTEM ON TAPE.

C-14 The procedures for saving an operating system on magnetic tape are as follows:

a. Mount a blank tape with a write ring inserted on the tape drive. Set the tape to the load point and put the tape drive on-line.

b. Make sure all system activity is terminated.

c. Transfer to the following file:

```
:TR,OSTAPE::40

*****  SAVING LU 2 ...
VERIFYING
 256 TRACKS SAVED
LSAVE:  STOP  0077

*****  SAVING LU 3 ...
VERIFYING
 256 TRACKS SAVED
LSAVE:  STOP  0077

*****  SAVING LU 40 ...
VERIFYING
 400 TRACKS SAVED
LSAVE:  STOP  0077

***** OPERATING SYSTEM SAVE COMPLETE
```

C-15 OFF-LINE RESTORE OF OPERATING SYSTEM.

C-16 The operating system (LU's 2, 3 and 40) can be restored from magnetic tape via the programs GACUP and !DISK. All system activity must be terminated for these programs to run. GACUP will load the off-line utility !DISK into memory. !DISK may also be loaded into memory via magnetic tape. !DISK will restore the operating system that was saved via LSAVE (see paragraph on saving an operating on system tape).

C-17 The following procedure is used to load !DISK via the program GACUP. User response is underlined.

:RU,GACUP (run from the system console)

*** GAC OPERATING SYSTEM UPDATE PROGRAM ***

!!! WARNING WARNING WARNING !!!

This program loads the OFF-LINE disc backup utility, and executes it at the SYSTEM CONSOLE.

All other system activity should be terminated.

THE RTE OPERATING SYSTEM IN MEMORY WILL BE DESTROYED.

DO YOU WANT TO ABORT THE RTE OPERATING SYSTEM?

YES

RTE ABORTED AND !DISK LOADED.

REFER TO RTE-IVB MANUAL FOR INSTRUCTIONS

!DISK is now loaded into memory. See below for responses to the !DISK program.

C-18 The following procedure is used to load !DISK from magnetic tape.

a. On the Magnetic Tape Drive Unit, perform the following:

   (1) Mount the Operating System Restore Program Magnetic Tape.

   (2) When tape is threaded, depress the LOAD Button.

   (3) When the LOAD Lamp illuminates, depress the ON LINE Button.

b. On the Computer, perform the following:

   (1) Select S register and then press CLEAR DISPLAY.

   (2) Enter 141614 (octal) and then press STORE.

   (3) Press PRESET, IBL/TEST, PRESET, RUN:

      a. Tape advances on Magnetic Tape Drive Unit.

      b. Computer Halts (RUN Light extinguished)

      c. T Register displays 102077 (octal).

   (4) Select S Register and then press CLEAR DISPLAY.

   (5) Enter 000014 (octal) and then press STORE.

(6) Select P Register and then press CLEAR DISPLAY.

(7) Enter 000002 (octal) then press STORE.

(8) Press the PRESET and RUN Switches.

c. The following display appears on the CRT:

```
DISK BACK UP UTILITY REV. 2026 800501
     LU    EQT    S.C.  S.CHNL       DRIVER
      1      3     14      0       DVR 0 CONSOLE
      4      4     13      0       DVA32 I.C. DISCS
      5      2     11      0       DVR32 13037 DISCS
      8      5     16      0       DVR23 MAG TAPE

SEL. CODE OF TBG = 10
ENTER SELECT CODE FOR DVR32, DVA32:
```

d. On the Keyboard, enter 11, 0 (RETURN).

e. The following display appears on the CRT:

TASK?

f. On the Magnetic Tape Drive Unit, perform the following:

(1) Rewind and remove the OPS Restore Program Magnetic Tape.

(2) Mount the Operating System Magnetic Tape.

(3) When tape is threaded, depress the LOAD Button.

(4) When the LOAD Lamp illuminates, depress the ON LINE Button.

g. On the Keyboard/CRT perform the following:

(1) Type in RE (RETURN) [For restore of LU2].

(2) The following display appears on the CRT:

LSAVE,1,2,8, VE, op sys
Tape #1
OK?

(3) Type in YES (RETURN).

(4) The Mag Tape moves forward and the following display appears on the CRT:

RESTORE TO 13037 DISC UNIT 0
DONE
TASK?

(5) Type in RE (RETURN) [For restore of LU3].

(6) The Mag Tape moves forward and the following display appears on the CRT:

```
LSAVE,1,3,8,VE, op sys
Tape #1
OK?
```

(7) Type in YES (RETURN).

(8) The following display appears on the CRT:

```
RESTORE TO 13037 DISC UNIT 0
DONE
TASK?
```

(9) Type in RE (RETURN) [For restore of LU40].

(10) The following display appears on the CRT:

```
LSAVE,1,40,8, VE, op sys
TAPE #1
OK?
```

(11) Type in YES (RETURN).

(12) The following display appears on the CRT:

```
RESTORE TO 13037 DISC UNIT 0
DONE
TASK?
```

(13) The computer is now ready for boot-up.

APPENDIX D

FILE MANAGER COMMANDS

D-1 INTRODUCTION.

D-2 The File Manager is used to manage user generated files. It responds to commands entered via the keyboard. This section will discuss the File Manager Commands. (Section IV discussed File Manager Organization).

D-3 File Manager Commands are run in a multi-terminal environment. This is explained as follows: RTE 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 can also be performed from any of the multiple terminals.

D-4 FMGR OPERATOR COMMANDS.

D-5 The program FMGR is a background task which is run from the system console or from a remote terminal in a multiterminal environment. FMGR allows an operator to perform the following basic functions:

a. Control FMGR by sending messages to the console or list device, requesting error explanations, changing the log or list devices or error severity.

b. Create and maintain files, both disc and non-disc, including maintenance of the file directory.

c. Keep track of the disc cartridge on which files are placed, including maintenance of the cartridge directory.

d. Transfer data or programs between files creating new files as needed.

D-6 INTERRUPTING FMGR.

D-7 Most FMGR Command processing can be interrupted with the RTE system command:

*BR, FMGR

D-8 If the FMGR is at a convenient breakpoint, command operation stops and the message:

:FMGR 000

is printed on the log device (usually the system console). If a job is being processed, the BR Command terminates the job and returns control to FMGR.

D-9 FMGR COMMANDS.

D-10 The FMGR commands are listed in Table D-1. This table presents the commands in the same functional groups in which they are described in this section.

Table D-1. FMGR Commands

FUNCTIONAL GROUP	COMMAND	FUNCTION
FMGR operation	??	Request error code explanation
	EX	Terminal FMGR; return control to RTE
	LL	Change list device
	CN	Control Functions (Non-Disc Devices)
File creation and manipulation	CR	Create a disc file
	PU	Purge a file
	ST	Store data in a device or file; create a file
	LI	List contents of a file
	RN	Rename a file
	DU	Transfer Data to an Existing File
Program file manipulation	RU	Execute program
	TR	Transfer Control to a File
FMGR Cartridge manipulation	CL	List cartridge directory
	DL	List file directory
	PK	Pack cartridge
	CO	Copy all files from cartridge to cartridge

D-11 COMMAND STRUCTURE.

D-12 Each command is specified by a mnemonic code consisting of at least two letters to indicate the operation to be performed. Depending on the command, parameters may be entered to further specify the command operation. Commands allow more than two characters in the command code, but only the first two characters are significant. For example, STORE can be specified, but ST is always sufficient.

D-13 The following syntax conventions are used in this manual to specify command format.

UPPER-CASE BLOCK	Literals that must be specified as shown; if underlined, the letters may be omitted.
lower-case italics	Type of information to be supplied by the user; most parameters are in this form.

[,parameter]	Optional parameters are enclosed in brackets; FMGR supplies a default value if omitted.
parameter 1 parameter 2 parameter 3	One and only one of the stacked parameters may be specified.
[parameter 1 parameter 2 parameter 3]	All bracketed parameters are optional;if all are omitted, FMGR supplies default value, or only one may be specified.
[,param1[,param2]]	Series of optional parameters; the last parameter may be omitted with no indication; embedded parameters must be indicated by a comma when omitted.
...	Ellipsis indicate that the previous parameter or series of bracketed parameters can be repeated.

To illustrate,

If the format is:	Enter:	
DL [,cartridge [,security]]	DL	default values supplied for parameter
	DL,,SC	position of parameter held by comma
	DL,2	last parameter omitted
LIST,namr [,SOURCE ,BINARY ,DIRECTORY]	LIST,2,DIRECTORY	use full names
	LI,2,D	use abbreviations for identical effect
	LIST,FILA	default parameter supplied

D-14 PARAMETER SYNTAX RULES.

D-15 A parsing routine checks every parameter specified in a FMGR command according to the following syntax rules:

a. The first parameter is separated from the command code by a comma (,) or a colon (:). Subsequent parameters are separated from each other by commas.

b. Subparameters are legal in the first two parameters. Subparameters are separated from each other by a colon (:). The first two subparameters may be ASCII or numeric, the rest must be numeric.

c. Blanks are ignored on either side of a delimiter or the command code.

d. Parameters are first assumed to be numeric, but if the parameter fails to convert, it is treated as ASCII unless it is a number immediately followed by B, G, P, or S or is preceded by a plus (+) or minus (-) sign.

e. Numeric parameters observe the following rules:

A leading plus sign (+) is ignored; a number is assumed to be positive unless preceded by a minus sign (-).

A number followed by the letter B is octal.

f. ASCII parameters are parsed to a maximum of six characters; only the first six characters are interpreted. If fewer than six, the parameter is padded with trailing blanks (octal code 40, the ASCII space).

g. The total length of all parameters in a single command must be less than:

128 - (8 times the parameter number)

h. The maximum number of parameters in one command is 14.

i. Comments may be entered following the last parameter as long as they do not replace an omitted optional parameter. They are subject to the length and number restriction on all parameters, but like messages are not interpreted.

D-16 NAMR PARAMETER.

FORMAT:

file name[:security[:cartridge[:file type[file size[:record size]]]]]

namr =

logical unit number

PURPOSE

Used to identify a file or device in a FMGR command. It uses subparameters and may appear only as the first or second parameter.

D-17 Unless specifically noted, each subparameter has a default value of zero. This value is selected so that, as closely as can be predicted, it provides the most general case. This means that in many cases, all subparameters can be omitted and the file be completely specified by name alone.

D-18 NAMR SUBPARAMETERS.

a. file name - 6-character ASCII file name, restricted as follows:

(1) only printable characters, space through _ (or ←)

(2) plus (+), minus (-), colon (:), or comma (,) not allowed

(3) first character must not be blank (space) or a number

(4) embedded blanks not allowed

(5) must be unique to FMGR cartridge.

b. logical unit - positive integer specifying logical unit number of non-disc device

c. security - positive or negative integer or 2 ASCII characters representing a positive integer; range is from -32767 through 32767; security may be:

(1) zero - file is unprotected (default)

(2) +integer - write protected; may be read with any security or none; may be written only with correct code or negative (2's complement) of correct code.

(3) -integer - file is fully protected; may be referenced only with correct negative code.

d. cartridge - positive or negative integer or 2 ASCII characters representing a positive integer; range is from -32767 through 32767; used to identify FMGR disc cartridge, it may be:

(1) zero - first available cartridge that satisfies the request is used; (default)

(2) +integer - cartridge reference number (CR) by which the cartridge is identified

(3) -integer - logical unit number (LU) associated with the cartridge. This is the same number as (2) except for a negative sign.

e. file type - Positive integer in range 0 through 32767. Files are a collection of information logically organized into records. They can be stored on disc or they may reference non-disc peripheral devices by name. The information in files may be programs or the data used by the programs. Data may be in binary or ASCII code. Programs may be in the form of ASCII source code, or binary code in either relocatable or absolute form. Programs may also be in memory-image form, a form used by RTE for programs ready to be executed.

- Eight file types are defined by the file system. Additional types may be defined by the user. Only the first four types differ in format; all subsequent types differ only in the type of data the file system expects the file to contain. The file types may be divided into three categories as shown in Table D-2. The first category contains one type, type zero. This type includes all non-disc devices defined as file types, types 1 and 2. These fixed-length record files are used for quick random access. The remaining file types all belong to the third category of files with variable-length records designed for sequential access. These files may be extended automatically as needed; files in the first two categories may not be extended.

Table D-2. Categories of File Types

CATEGORY	TYPE	DESCRIPTION
Fixed-length, random access, non-extendable	1	Fixed-length 128 word record files
	2	Fixed user-defined record length files
Variable length, sequential access, automatic extents	3	Variable-length records; any data type
	4	Source program file; ASCII
	5	Object program file; relocatable binary
	6	Executable program file; memory-image code
	7	Absolute binary
	8-32767	User defined data format

f. file size - This number indicates the number of 128 word blocks to be allocated to the file at creation time.

g. record size - decimal number of words in range 1 through 32767. Applies only to type 2 files; type 1 files use 128-word records, other types use variable length records.

D-19 NAMR EXAMPLES.

a. 10 - logical unit 10

b. 20B - logical unit 16 (octal 20)

c. $XYZ:AA - file named $XYZ is write protected by ASCII code AA (040501 octal or 16705 decimal)

D-20 FMGR OPERATION AND RUN COMMAND.

D-21 To use FMGR interactively, you simply run the program from a terminal. When it responds with the colon prompt, you may enter any command. To run in batch mode, you also run the program from a terminal but specify that command input is to be from an input device such as the paper tape reader or from a disc file. No prompt is issued. The program FMGR expects a colon to precede each command entered from a non-interactive device or disc.

D-22 When you run FMGR, it assumes default values for the devices used for command input, to log errors and to list output. These values all assume interactive mode.

D-23 When you enter each command at a terminal, it is echoed back to you at the terminal. It appears as if you were typing the command on the terminal display, but actually the command is sent to FMGR which then displays (echoes) the command. Command echo can be suppressed, and it is good practice to suppress the echo when command input is from a non-interactive device.

D-24 During FMGR operation, messages can be issued to the terminal, to the log device if it is different from the terminal, or to the list output device. Each type of message uses a different command. The message to the log device also causes a pause in job processing so that you may interact with FMGR.

D-25 FMGR operation is terminated with the EX command.

D-26 RUNNING PROGRAM FMGR.

D-27 RU, FMGR.

FORMAT:

RU,FMGR

PURPOSE

Requests FMGR from the RTE system console.

D-28??

FORMAT:

:??,error#

Parameter:

error # Error code whose explanation is requested; if omitted, the explanation of the current error is issued; if two error codes were sent, the explanation usually refers to the first number only. If error# is 99, a list of all FMGR error codes and their explanations is printed at the list device.

PURPOSE

Requests a brief description of an error code during interactive operation.

D-29 Once FMGR is running, it assumes control and if a command cannot be understood (input error) or has caused a recognized problem, an error message is printed in the form:

FMGR nnn

where nnn is a three-digit number.

D-30 In some cases, when an error code is issued, it is followed automatically by additional information. This may consist of the line in which the error occurred up to the point where the error was detected. Or it may be a second FMGR error code. Any error code explanation can be requested by entering the code number as the error# parameter. Be sure to include the comma separator or the current error will be explained.

See Appendix E for a list of FMGR error codes and explanations.

D-31 EX.

FORMAT:

:EXIT

PURPOSE

Terminates the program and returns it to RTE control when finished using FMGR.

D-32 In response to EX, the FMGR sends the message:

$END,FMGR

to the log device unless a non-zero severity code inhibits the message.

D-33 LL.

FORMAT:

:LL,namr

Parameters

namr   New list device, may be either a file or a logical unit number.

PURPOSE

Changes the list device currently assigned to FMGR.

D-34 namr may refer to any existing file or logical unit. It should be a device allowing output; an attempt to list on an input device terminates FMGR and issues the system error code I007.

Examples

1. *RU,FMGR
   :LL,4   Change the list device from default logical unit 1

(console) to logical unit 4 (paper tape punch)

.
.
.

```
:LL,6                   Subsequently change it to logical unit 6.
```

2.
```
*RU,FMGR
:LL,LISTF::13           Change list device to list output on existing file
                        LISTF;
                        Specify CR to insure that list is sent to correct file.
```

D-35 CN AND CT

FORMAT:

```
CN,namr,function
CT,name,function
```

namr
Logical unit number of the device to be controlled or its type 0 file name previously defined in a CR command; default is 8 (recommended logical unit for magnetic tape). Range of logical unit numbers is from 1 to 63.

function
Control function to be performed on non-disc device.

PURPOSE:

CN controls non-disc devices. The magnetic codes are:

RW -------- Rewind (default for mag tape, terminal cartridge tape unit, and mass storage devices).
EO -------- End-of-file
TO -------- Top-of-form (default for line printer and terminal CRT).
FF -------- Forward Space File
FR -------- Forward Space Record
BR -------- Backspace Record
LE -------- Leader (default for paper tape punch).

EXAMPLE:

To rewind a magnetic tape:

```
:CN,8,RW
```

CT Used to issue control requests to an interactive terminal.

function
Control function to be performed on interactive device. Default is to enable terminal (octal function code 20B).

11B ---Space down a specified number of lines (used in conjunction with subfunction parameter).
20B ---Enable terminal.
21B ---Disable terminal.
22B ---Set time-out for terminal (used in conjunction with subfunction parameter).

D-36 FILE CREATION AND MANIPULATION.

File creation for disc files means that the file is given a name, a file directory entry, and is assigned an area on disc. For non-disc files, creation means associating a peripheral device with a file name so that it can be treated as a file in subsequent operations.

D-37 Files may be created in a variety of ways. Probably the most commonly used command, STORE, creates a file and transfers data to the file in the process. The CREATE command simply creates a disc file with no data. Special files containing programs may also be created with FMGR commands. See Table D-2 for the different types of FMGR files.

D-38 The transfer of data is accomplished with the STORE command. This command allows the transfer of data between peripheral devices or between a device and a disc file or between disc files.

D-39 CR.

FORMAT:

:CREATE,namr

Parameters

namr	File descriptor; must not be a logical unit number; omitted subparameters default to zero; file type and file size must be specified as greater than zero.

PURPOSE

Creates a disc file. No data is transferred to the file.

D-40 When a disc file is created, an entry is made at the end of the file directory on the cartridge to which the file is allocated. If the subparameter cartridge is specified, the file is allocated to that cartridge; otherwise, it is sent to the first cartridge found with enough room starting at the head of the cartridge directory on the system disc. If a file with the given name already exists on the first cartridge with enough space, an error -002 is issued.

Example: MYFILE is type 4;

:CR,MYFILE:-25:100:4:50	it has a security code of -25 (read and write are restricted to users knowing the code); it is allocated to a cartridge with CR = 100 it is a type 4 file it has a size of 50 blocks

D-41 PU.

FORMAT:

:PURGE,namr

Parameters

namr — File descriptor; enter file name and, optionally, security code and cartridge reference.

PURPOSE

Removes a file and its extents.

D-42 If a file is protected by a security code it cannot be purged unless the correct code is entered.

D-43 If a label is specified, that cartridge is searched for the file to be purged; otherwise, cartridges are searched and the first file found with the correct file name is purged.

Examples

1. :PU,AA — purge the first file found named AA

2. :PU,CC:55:100 — purge file CC protected by security code 55 on cartridge 100

D-44 ST and DU.

FORMAT:

:STORE,namr1,namr2 or DUMP,namr1,namr3

Parameters

namr1 — File name of existing file or a logical unit number; data is transferred from namr1. (See namr description).

namr2	File name or logical unit to which data is transferred from namr1; if a file name, the file is created using the last two namr subparameters if supplied.
namr3	Name of existing file or LU to which data is transferred.

PURPOSE

Transfers data from a file or logical unit, to a disc file or logical unit; the receiving file is created by the command unless it is a logical unit.

Note

Files are transferred record by record; records longer than 128 words are truncated.

namr1 can be a created file (disc or non-disc) or a logical unit number defined for the system;

namr2 can be a logical unit number or a disc file name; it may not name a type 0 file. The file type of namr2 is the same as the file type of namr1 if namr1 is a disc file.

If namr1 is not a disc file, the file type of namr2 is based on the record format of namr1.

Examples

1. Transfer contents of disc file XYZ to new file AA:

```
:ST,XYZ,AA
```

2. Copy a file from cartridge 2 to cartridge 17:

```
:ST,XYZ::2,XYZ::17       file XYZ on cartridge 2 is not affected
```

3. Duplicate a paper tape:

```
:ST,5,4           the contents of tape on LU 5 are punched on a
                  new tape
```

4. Transfer the contents of disc file XYZ to an existing file ABC

```
:DU,XYZ,ABC
```

D-45 LI.

FORMAT:

:LIST,namr

Parameters

namr	File name or logical unit number; (refer to namr description) if file is protected by a negative security code, it must be specified; if cartridge reference number is included, that cartridge is searched for the file name, otherwise, the first file found with that name is listed.

PURPOSE

Lists the contents of a file, file directory information, or data stored on a logical unit on the list device.

Note

The LI command lists the specified file record by record.

D-46 HEADINGS.

D-47 If namr is a file, the listing is headed by:

```
file name    T = file type    IS ON CR cartridge    USING file size
                        BLKS R = record size
```

where the italicized words are replaced by the actual values in the file directory for the file (see examples).

D-48 If namr is a logical unit, then a brief heading is printed with asterisks replacing the file name:

```
****** T=00000 IS ON LU nn
```

where nn is the logical unit number.

Example

```
:LI,AA

AA T=00003 IS ON CR0002 using 1 BLKS R = 128

0001 first record of file AA
0002 second record of file AA
.
.
.
.
.
```

nnnn last record of file AA

D-49 RN.

FORMAT:

:RN,namr,name

Parameters

namr	File name and, optionally, security code and cartridge reference number of existing file; (see namr description)
name	New file name to replace existing file name

Note

Subparameters of namr may not be changed

PURPOSE

Renames an existing, but closed, disc file; none of the file characteristics except the name are changed.

D-50 The new name must be unique to the FMGR cartridge to which the existing file is allocated. If the file was created with a non-zero security code, then the namr in this command must include that code. If a cartridge reference number is included in the namr, then only that FMGR cartridge is searched. If the CR number is omitted, all mounted cartridges are searched and the first file found with the corresponding namr is renamed. The search starts with the first cartridge in the cartridge directory.

Examples

1.	:RN,MYFILE:-25:100,MF	search for MYFILE on cartridge number 100 and, if the security code is correct, change its name to MF.
2.	:RN,URFILE,FILEA	search cartridges for first file named URFILE and change its name to FILEA; the file is not protected by a security code.

D-51 RU.

FORMAT:

:RU,program

Parameters:

program · Five character name of program to be executed.

PURPOSE

Searches for and executes the named program.

D-52 When RU is used, a search is made of the system ID segments for the named program. If found, the program is executed. If not found, a search is made of the FMGR directories for the named file. If such a file is found, an ID segment is built in memory, and the program is executed.

D-53 TR (Transfer Control to a File)

Allows transfer of control to a file

FORMAT:

:TRANSFER,namr

---

namr
Identifies file to which TR transfers; if omitted, TR returns control to the namr of a previous transfer; up to 10 transfers can be nested - a stack of return pointers is saved.

D-54 FMGR CARTRIDGE MANIPULATION.

In the File Manager Program, disc manipulation is performed in terms of FMGR cartridges. An FMGR cartridge is a logical area on a disc consisting of an area for file storage and a directory of the files stored. Each cartridge is a physically contiguous area on the disc identified in FMGR calls and commands by a unique cartridge reference number. It also has an ASCII identifier and must be associated with a unique logical unit number.

D-55 CL.

FORMAT:

:CL

PURPOSE

Requests a list of all cartridges in cartridge directory.

D-56 The cartridge list is issued to the list device, normally LU1, the console. This device may be changed with LL. The list contains the following categories:

LU	Logical unit number of the cartridge.
LAST TRACK	Last track assigned to the FMGR on that cartridge.
CR	Cartridge reference number.
LOCK	Name of program locking the cartridge; blank if not locked.

Example

:CL   request from input device

LU	LAST TRACK	CR	LOCK	
41	399	41		
42	399	42		output to list device
2	255	2.		
3	150	3		

D-57 DL,

FORMAT:

:DL[,cartridge[,master security]]

Parameters

cartridge	Cartridge identifier; positive for cartridge reference number, negative for logical unit number; if omitted or zero, the directories of all mounted cartridges are listed.
master security	Code assigned to the system at initialization; if correctly specified, directory list includes file security code and track and sector address for each file (long list).

PURPOSE

Each mounted cartridge has on its last tracks a directory for the cartridge. The directory contains an entry describing the cartridge followed by an entry describing each file on the cartridge. The DL command requests a listing of this directory.

D-58 The directory list is provided in two formats: a short list and a long list (see Figure D-1). Both lists have the same header information describing the cartridge itself; they differ in the file information provided. The long list includes a file security code for each file and the track and sector address for the file.

D-59 PK.

FORMAT:

:PK,[cartridge]

Parameters

cartridge	Cartridge identifier; a positive reference number or, if negative, the logical unit number of the FMGR cartridge to be packed; if omitted, all mounted cartridges are packed.

PURPOSE

Recovers the tracks assigned to purged files and closes any gaps between files.

D-60 When PK executes, it moves files into empty spaces left from purging, if possible, and updates the file addresses in the file directory. When all files are packed, it then packs the directory removing any entries for purged files.

D-61 CO.

FORMAT:

:COPY,cartridge1,cartridge2

Parameters

cartridge1	Cartridge reference number of mounted cartridge containing files to be copied; if negative, identifies cartridge logical unit number.
cartridge2	Cartridge reference number of mounted cartridge to which files are to be transferred; if negative, identifies cartridge logical unit number.

PURPOSE

Copies all files, currently on a mounted cartridge, to another mounted cartridge.

D-62 All files transferred should have unique names. If a file on cartridge2 has the same name as a file being transferred from cartridge1, the file is not transferred and an informative message is sent to the log device. As each file is copied, its name is displayed on the log device.

D-63 The files being copied are not affected by the copy. If files already exist on the cartridge to which files are being copied (cartridge2), these files are not affected; the copied files follow the existing files. Entries for the copied files are added to the end of the file directory on cartridge2. If there are any entries for purged files in this directory, entries for the copied files may be interspersed with entries for existing files. To know where the new file entries are placed, request a directory list with the DL command.

Example

Assume files A, B, C, and D on cartridge1 are to be copied to cartridge2, and a file C already exists on cartridge2:

:CO,LA,LB	where LA is cartridge1 and LB is cartridge2
A	system prints file names as they are copied
B	
C	
FMGR-002	message indicating C is a duplicate name; it is not copied
D	

:CO,14,22	Where LU 14 is a disc LU and LU 22 is also a disc LU. The contents of LU 14 (excluding duplicate names) will be transferred to LU 22.

Figure D-1. Directory List Types

SHORT LIST

Cartridge reference number

Command

Cartridge information label

Next available track

Next available sector

No. of Sectors/Track

Last FMGR track on this disc

No. of Directory tracks

DL,2

CR=00002

ILAB-SYSDIS NXTR=0133 NXSEC=074 # SEC/TR=096 LAST TR=0202 # DR TR=02

NAME	TYPE	SIZE/LU	OPEN TO
MT	00000	08	
REWND	00000	08	FMGR
EOF	00004	00001	
TAPNUM	00002	00128	FMGR

File Name

File Type

Number of blocks used by file, or a logical unit number if is type 0.

File is open to this program. Extents are indicated by a plus (+) sign preceding the extent number. Exclusive opens are indicated by a minus (-) sign following the program name.

LONG LIST

DL,2,S

CR=00002

ILAB-SYSDIS NXTR=0133 NXSEC=074 # SEC/TR=096 LAST TR=0202 # DR TR=02

NAME	TYPE	SIZE/LU	SCODE	TRACK	SEC	OPEN TO
MT	00000	08	00000			
REWND	00000	08	00000			FMGR
EOF	00004	00001	00000	0100	000	
TAPNUM	00002	00128	12345	0100	002	FMGR

File security code

Track and sector address

APPENDIX E

FMGR ERROR CODES

**

FMGR-102

ILLEGAL D.RTR CALL SEQUENCE
A LOCK WAS NOT REQUESTED FIRST OR THE FILE WAS NOT OPENED EXCLUSIVELY. POSSIBLY AN OPERATOR ERROR, SUCH AS REMOVING A CARTRIDGE WITHOUT DISMOUNTING IT FIRST.

**

FMGR-101

ILLEGAL PARAMETER IN D.RTR CALL
POSSIBLY AN OPERATOR ERROR. RECHECK THE PREVIOUS ENTRIES FOR ILLEGAL OR MISPLACED PARAMETERS.

**

FMGR-099

DIRECTORY MANAGER EXEC REQUEST ABORTED
AN EXEC REQUEST MADE BY D.RTR WAS ABORTED. MAKE SURE THAT ALL DISCS BEING ACCESSED ARE UP. NOTIFY SYSTEM MANAGER.

**

FMGR-048

SPOOL NOT INITIALIZED OR SMP CANNOT BE SCHEDULED

**

FMGR-046

GREATER THAN 255 EXTENTS
ATTEMPT TO CREATE EXTENT 256. MAKE FILE SIZE OF MAIN LARGER.

**

FMGR-041

NO ROOM IN SST

**

FMGR-040

LU NOT FOUND IN SST
TRYING TO ACCESS AN LU THAT IS NOT IN YOUR SST. USE THE SL COMMAND TO ADD THE LU TO THE SST.

**

FMGR-039

CONFLICT IN SST DEFINITION

**

FMGR-038

ILLEGAL SCRATCH FILE NUMBER

**

FMGR-035

ALREADY 63 DISCS MOUNTED TO SYSTEM
AN ATTEMPT WAS MADE TO MOUNT A DISC WHEN THERE ARE ALREADY 63 DISCS MOUNTED. A DISC WILL HAVE TO BE DISMOUNTED BEFORE A NEW ONE MAY BE MOUNTED.

**

FMGR-034

DISC ALREADY MOUNTED.
AN ATTEMPT WAS MADE TO MOUNT A DISC THAT IS ALREADY MOUNTED ON THE CARTRIDGE LIST. EITHER DISMOUNT THE DUPLICATE DISC OR MOUNT A DIFFERENT ONE.

**

FMGR-033

NOT ENOUGH ROOM ON CARTRIDGE
AN ATTEMPT WAS MADE TO ACCESS A CARTRIDGE WHICH HAS NO MORE ROOM. TRY USING ANOTHER CARTRIDGE OR DECREASE THE FILE SIZE.

**

FMGR-032

CARTRIDGE NOT FOUND
AN ATTEMPT WAS MADE TO ACCESS A CARTRIDGE THAT CANNOT BE FOUND IN THE CARTRIDGE LIST. CHECK THE CARTRIDGE NUMBER FOR CORRECTNESS.

**

FMGR-030

VALUE TOO LARGE FOR PARAMETER

**

FMGR-026

QUEUE FULL OR MAX PENDING SPOOLS EXCEEDED
THE SPOOL QUEUE IS FULL OR THE MAXIMUM NUMBER OF SPOOLS PENDING HAS BEEN EXCEEDED. THE JOB MUST BE RE-RUN WHEN THE SPACE BECOMES AVAILABLE.

..
FMGR-025

NO SPLCON ROOM
THE SPLCON IS FULL. THIS ERROR MAY OCCUR WHEN THE SPOOL SYSTEM IS COMPETING WITH PROGRAMS USING THEIR OWN SPOOLING FILE AND RUNNING OUTSIDE OF BATCH.

..
FMGR-024

NO MORE BATCH SWITCHES
THE LU SWITCH TABLE IS FULL. THE SIZE OF THE SWITCH TABLE SPECIFIED AT SYSTEM GENERATION IS INADEQUATE. NOTIFY THE SYSTEM MANAGER OF THIS CONDITION.

..
FMGR-023

NO AVAILABLE SPOOL FILES
ALL SPOOL FILES ARE CURRENTLY BEING USED. RE-RUN THE JOB AFTER A SPOOL FILE BECOMES AVAILABLE.

..
FMGR-022

NO AVAILABLE SPOOL LU'S
ALL SPOOL LOGICAL UNITS ARE CURRENTLY UNAVAILABLE. RE-RUN THE JOB AFTER A SPOOL LU BECOMES AVAILABLE.

..
FMGR-021

ILLEGAL DESTINATION LU
THE LU SPECIFIED WAS NOT ALLOCATED BY GASP. TRY AGAIN USING A LU ALLOCATED BY GASP.

..
FMGR-020

ILLEGAL ACCESS LU
1. THE LOGICAL UNIT NUMBER SPECIFIED IN THE LU OR CS COMMAND WAS NOT A POSITIVE LOGICAL UNIT NUMBER. RE-ENTER THE CORRECTED COMMAND. OR
2. THERE IS AN LU ENTRY IN THE CARTRIDGE LIST THAT DOES NOT POINT TO A DISC DEVICE. THIS HAPPENED BECAUSE AFTER THE DISC WAS MOUNTED THE LU COMMAND WAS USED TO DO A LOGICAL UNIT SWITCH ON THE DEVICE. SWITCH THE LU BACK TO ITS DISC DEFINITION. IF DESIRED, DISMOUNT THE DISC. THE LU CAN THEN BE SWITCHED TO A NON-DISC DEVICE.

**

FMGR-019

ILLEGAL ACCESS ON A SYSTEM DISC
AN ATTEMPT WAS MADE TO WRITE ON A SYSTEM DISC. THE SYSTEM MANAGER IS THE ONLY USER THAT HAS THIS CAPABILITY.

**

FMGR-018

ILLEGAL LU; LU NOT ASSIGNED TO SYSTEM
ATTEMPT TO ACCESS AN LU THAT IS NOT ASSIGNED TO THE SYSTEM.

**

FMGR-017

ILLEGAL READ/WRITE ON TYPE 0 FILE
AN ATTEMPT WAS MADE TO READ, WRITE, OR POSITION A TYPE 0 FILE THAT DOES NOT SUPPORT THE OPERATION. CHECK THE FILE PARAMETERS OR THE NAMR.

**

FMGR-016

ILLEGAL TYPE 0 O° SIZE=0
ONE OF THE FOLLOWING OCCURED:
1) THE WRONG FILE TYPE WAS SPECIFIED,
2) AN ATTEMPT WAS MADE TO CREATE OR PURGE A TYPE 0 FILE, OR
3) THE SIZE SPECIFIED WAS ZERO.
CHECK THE SIZE AND TYPE PARAMETERS.

**

FMGR-015

ILLEGAL NAME
THE FILE NAME DOES NOT CONFORM TO THE SYNTAX RULES. CORRECT THE NAME AND RE-ENTER THE COMMAND.

**

FMGR-014

DIRECTORY FULL
THERE IS NO MORE ROOM IN THE FILE DIRECTORY. PURGE ANY UNUSED FILES AND PACK THE DISC IF POSSIBLE. OTHERWISE, TRY ANOTHER CARTRIDGE.

**

FMGR-013

DISC LOCKED
THE CARTRIDGE SPECIFIED IS LOCKED. INITIALIZE THE CARTRIDGE IF IT WAS NOT INITIALIZED, OTHERWISE KEEP TRYING.

**
FMGR-012

EOF OR SOF ERROR
AN ATTEMPT WAS MADE TO READ, WRITE, OR POSITION A FILE BEYOND THE FILE BOUNDARIES. CHECK THE RECORD POSITION PARAMETERS. THE RESULTS DEPEND ON THE FILE TYPE AND THE CALL.

**
FMGR-011

DCB NOT OPEN
AN ATTEMPT WAS MADE TO ACCESS AN UNOPENED DCB. USE THE CREATE OR OPEN CALL TO OPEN THE DCB AND CHECK FOR ERRORS.

**
FMGR-010

NOT ENOUGH PARAMETERS
ONE OR MORE OF THE REQUIRED PARAMETERS WERE OMITTED FROM THE CALL. ENTER THE REQUIRED PARAMETERS.

**
FMGR-009

ATTEMPT TO USE APOSN OR FORCE TO 1 A TYPE 0 FILE
A TYPE 0 FILE CANNOT BE POSITIONED WITH APOSN OR BE FORCED TO A TYPE 1 FILE. CHECK THE FILE TYPE.

**
FMGR-008

FILE OPEN OR LOCK REJECTED
AN ATTEMPT WAS MADE TO OPEN A FILE THAT WAS ALREADY OPENED EXCLUSIVELY OR WAS ALREADY OPENED TO EIGHT PROGRAMS, OR THE CARTRIDGE CONTAINING THE FILE IS LOCKED. USE THE CL OR DL COMMAND TO LOCATE THE LOCK. IF THE FILE IS BEING PACKED, CHECK TO SEE IF SPOOLING IS SHUT DOWN.

**
FMGR-007

ILLEGAL SECURITY CODE OR ILLEGAL WRITE ON LU2 OR 3
1. AN ATTEMPT WAS MADE TO ACCESS A FILE WITHOUT SPECIFYING THE SECURITY CODE OR WITH THE WRONG SECURITY CODE. FIND OUT THE CORRECT CODE AND USE IT OR DO NOT ACCESS THE FILE. OR
2. AN ATTEMPT WAS MADE BY A SESSION USER (NOT THE SYSTEM MANAGER) TO WRITE ON LU 2 OR 3. SESSION USERS DO NOT HAVE WRITE ACCESS TO LU 2 OR 3.

**

FMGR-006

FILE NOT FOUND
AN ATTEMPT WAS MADE TO ACCESS A FILE THAT CANNOT BE FOUND. CHECK THE FILE NAME.

**

FMGR-005

RECORD LENGTH ILLEGAL
AN ATTEMPT WAS MADE TO READ OR POSITION A FILE TO A RECORD THAT HAS NOT BEEN WRITTEN, OR TO WRITE AN ILLEGAL RECORD LENGTH ON AN UPDATE. CHECK THE FILE POSITION OR SIZE PARAMETER.

**

FMGR-004

MORE THAN 32767 RECORDS IN A TYPE 2 FILE
AN ATTEMPT WAS MADE TO CREATE A TYPE 2 FILE WITH TOO MANY RECORDS OR WITH A RECORD SIZE THAT IS TOO LARGE. CHECK THE SIZE PARAMETER.

**

FMGR-003

BACKSPACE ILLEGAL
AN ATTEMPT WAS MADE TO BACKSPACE A DEVICE (OR TYPE 0 FILE) THAT CANNOT BE BACKSPACED. CHECK THE DEVICE TYPE.

**

FMGR-002

DUPLICATE FILE NAME
A FILE ALREADY EXISTS WITH THE NAME SPECIFIED. REPEAT THE COMMAND WITH A NEW NAME OR PURGE THE EXISTING FILE.

**

FMGR-001

DISC ERROR
THE DISC IS DOWN. TRY AGAIN AND THEN REPORT THE PROBLEM TO THE SYSTEM MANAGER.

**

FMGR 000

BREAK
THIS IS AN INFORMATIVE MESSAGE ONLY. NO ERROR HAS OCCURRED.

**

FMGR 001

DISC ERROR - LU REPORTED
THE DISC ASSOCIATED WITH THE LU REPORTED IS DOWN. REPORT THE PROBLEM TO THE SYSTEM MANAGER.

**

FMGR 002

INITIALIZE LU 2!
THIS ERROR INDICATES A REQUEST FOR THE COMMAND TO INITIALIZE THE SYSTEM DISC (LU 2). ENTER THE INITIALIZE COMMAND.

**

FMGR 003

INITIALIZE LU 3!
THIS ERROR INDICATES A REQUEST FOR THE COMMAND TO INITIALIZE THE AUXILIARY DISC (LU 3). ENTER THE INITIALIZE COMMAND.

**

FMGR 004

ILLEGAL RESPONSE TO FMGR 002 OR FMGR 003
A COMMAND OTHER THAN AN INITIALIZE COMMAND WAS ENTERED IN RESPONSE TO EITHER A FMGR 002 OR FMGR 003 ERROR. ENTER THE APPROPRIATE INITIALIZE COMMAND.

**

FMGR 005

REQUIRED TRACK NOT AVAILABLE - RELATIVE TAT POSITION REPORTED
THE FIRST TRACK SPECIFIED IN THE INITIALIZE COMMAND IS NOT AVAILABLE. RE-ENTER THE INITIALIZE COMMAND WITH THE FIRST AVAILABLE TRACK REPORTED IN THIS MESSAGE.

**

FMGR 006

FMGR SUSPENDED
THE FILE MANAGER SUSPENDED ITSELF. READY THE DOWN DEVICE AND ENTER 'GO,FMGR'.

**

FMGR 007

CHECKSUM ERROR
A CHECKSUM ERROR OCCURRED WHEN READING A PAPER TAPE OR THE FILE BEING READ IS NOT BINARY (TYPE 5 OR 7). CHECK THE FILE TYPE.

..

FMGR 008

D.RTR NOT LOADED
THE PROGRAM D.RTR WAS NOT FOUND IN THE SYSTEM. LOAD D.RTR AS A PERMANENT PROGRAM.

..

FMGR 009

ID SEGMENT NOT FOUND
AN RP COMMAND WAS USED TO DEALLOCATE OR REASSIGN THE ID SEGMENT TO THE PROGRAM BEING RESTORED. THE SYSTEM LOOKS FOR A BLANK ID SEGMENT.

..

FMGR 010

INPUT ERROR
A SYNTAX ERROR IN THE STATEMENT OCCURRED. LOOK FOR A MISSING COLON (BATCH INPUT) OR EXTRA COLON (INTERACTIVE INPUT), AN UNDEFINED COMMAND, AN ERROR IN THE NAMR SUBPARAMETERS, A COMMAND THAT IS TOO LONG, ETC. RE-ENTER THE COMMAND. IF RECEIVED AFTER ENTERING AN ABORT COMMAND, THERE WERE NO ACTIVE JOBS.

..

FMGR 011

DO ´OF,XXXXX,8´ ON NAMED PROGRAMS
AN ATTEMPT WAS MADE TO PACK A DISC TO WHICH THE NAMED PROGRAMS ARE STILL ALLOCATED. ENTER EITHER ´RP,NAMR,PROGRAM´ OR ´OF,PROGRAM,8´ TO REMOVE THE NAMED PROGRAMS.

..

FMGR 012

DUPLICATE DISC LABEL OR LU
AN ATTEMPT WAS MADE TO MOUNT A CARTRIDGE THE SAME LABEL OR LOGICAL UNIT NUMBER OF A CARTRIDGE THAT IS ALREADY MOUNTED. RE-ENTER THE THE COMMAND WITH ANOTHER LABEL OR LU, OR DISMOUNT THE DUPLICATE CARTRIDGE.

..

FMGR 013

TR STACK OVERFLOW
MORE THAN 10 NESTED TR COMMANDS HAVE BEEN USED. CORRECT THE CODING.

..

FMGR 014

REQUIRED ID SEGMENT NOT FOUND
AN ID SEGMENT CANNOT BE FOUND FOR THE SPECIFIED PROGRAM. CHECK THE PROGRAM NAME OR LOAD THE PROGRAM. A BLANK ID SEGMENT CANNOT BE FOUND FOR A PROGRAM BEING RESTORED. ENTER AN 'OF' COMMAND TO RELEASE AN ID SEGMENT.

..

FMGR 015

LS TRACK REPORT
THIS IS AN INFORMATIVE MESSAGE TO REPORT THE LOGICAL UNIT NUMBER AND TRACK OF THE CURRENT LS AREA.

..

FMGR 016

FILE MUST BE AND IS NOT ON LU 2 OR LU 3
AN ATTEMPT WAS MADE TO RESTORE A PROGRAM FILE THAT IS NOT ON THE SYSTEM OR AUXILIARY DISC. MOVE THE FILE TO LU 2 OR LU 3 AND RE-ENTER THE COMMAND.

..

FMGR 017

ID SEGMENT NOT SET UP BY RP
IN ORDER FOR AN ID SEGMENT TO BE RELEASED BY A 'RP' COMMAND, IT MUST HAVE BEEN SET UP BY A 'RP' COMMAND. TRY USING 'OF,PROGRAM' TO RELEASE THE SPECIFIED PROGRAM.

..

FMGR 018

PROGRAM NOT DORMANT
AN 'RP,NAMR,PROGRAM' COMMAND WAS ATTEMPTED WHEN THE PROGRAM IS ACTIVE. ENTER 'OF,PROGRAM' AND THEN REPEAT THE 'RP' COMMAND.

..

FMGR 019

FILE NOT SET UP BY SP ON CURRENT SYSTEM
THE PROGRAM FILE BEING RESTORED HAD A PARITY ERROR, WAS NOT SET UP CORRECTLY, OR WAS NOT SET UP BY A 'SP' COMMAND IN THE CURRENT SYSTEM. RELOAD THE PROGRAM AND TRY AGAIN.

..

FMGR 020

ILLEGAL TYPE 0 FILE
AN ATTEMPT WAS MADE TO CREATE A TYPE 0 FILE ON A LOGICAL UNIT THAT IS NOT ASSIGNED IN THE SYSTEM. RE-ENTER THE COMMAND USING ANOTHER LOGICAL UNIT.

[illegible]

ILLEGAL DISC SPECIFIED
AN ATTEMPT WAS MADE TO COPY FILES TO OR FROM THE SAME DISC OR A DISC THAT IS NOT MOUNTED. MOUNT ANOTHER DISC OR USE ANOTHER ALREADY MOUNTED.

FMGR 022

COPY TERMINATED
[illegible] HAS BEEN TERMINATED AS A RESULT OF COPY ERROR. CHECK THE [illegible] AND THE SPECIFIED DISC.

FMGR 023

DUPLICATE PROGRAM NAME
THE PROGRAM BEING RESTORED IS ALREADY DEFINED IN THE SYSTEM. CHANGE THE NAME OF THE PROGRAM, ENTER 'OF,PROGRAM', OR RELEASE THE ID SEGMENT.

FMGR 041

[illegible]

[illegible]2

[illegible] SWITCHED

LU NOT FOUND IN SST

FMGR 044

NO MESSAGES WAITING
CALLER ISSUED A ME COMMAND BUT THERE WERE NO MESSAGES WAITING TO BE [illegible].

SESSION COMMAND ONLY
THE SPECIFIED COMMAND OPERATES ONLY IN THE SESSION ENVIRONMENT.

**
FMGR 046

INSUFFICIENT CAPABILITY
AN ATTEMPT WAS MADE TO EXECUTE A COMMAND THAT REQUIRES A HIGHER CAPABILITY LEVEL THAT THE CAPABILITY LEVEL DEFINED FOR THIS SESSION USER.

**
FMGR 047

SPOOL SET UP FAILED
THERE ARE NO AVAILABLE SPOOL FILES OR LOGICAL UNITS, OR THE LOGICAL UNIT TABLE IS FULL. YOU CAN TRY RUNNING THE JOB AGAIN, BUT IF THE ERROR IS FROM A LACK OF SPOOL LOGICAL UNITS OR THE LOGICAL UNIT TABLE BEING FULL YOU MUST RECONFIGURE.

**
FMGR 048

GLOBAL SET OUT OF RANGE
A GLOBAL WAS SPECIFIED OUT OF THE RANGE OF THE GLOBALS. CHECK THE PARAMETERS AND RE-ENTER THE COMMAND CORRECTLY.

**
FMGR 049

CAN´T RUN RP´ED PROGRAM
THE PROGRAM RESTORED FROM THE FILE DOES NOT EXECUTE. USUALLY THIS IS CAUSE BY ATTEMPTING TO RUN A SEGMENT OF THE SPECIFIED PROGRAM. CHECK THE PROGRAM.

**
FMGR 050

NOT ENOUGH PARAMETERS
LESS THAN THE REQUIRED NUMBER OF PARAMETERS WERE SPECIFIED. RE-ENTER COMMAND CORRECTLY.

**
FMGR 051

ILLEGAL MASTER SECURITY CODE
AN ATTEMPT WAS MADE TO RE-INITIALIZE A CARTRIDGE OR LIST FILES WITH AN INCORRECT MASTER SECURITY CODE. RE-ENTER THE COMMAND WITH THE CORRECT CODE.

**
FMGR 052

ILLEGAL LU IN RESPONSE TO 002 OR 003
AN ATTEMPT WAS MADE TO INITIALIZE THE FILE MANAGER USING A LOGICAL UNIT OTHER THAN LU 2 OR LU 3, OR AN ATTEMPT WAS MADE TO MOUNT A LOGICAL UNIT WHICH IS NOT A DISC CARTRIDGE. CHECK THE LU AND RE-ENTER THE COMMAND CORRECTLY.

**

FMGR 053

ILLEGAL LABEL OR ILABEL
THE SPECIFIED CARTRIDGE REFERENCE NUMBER OR CARTRIDGE ID IS ILLEGAL. THE CARTRIDGE REFERENCE NUMBER MUST BE A POSITIVE NON-ZERO INTEGER AND THE CARTRIDGE ID MUST BE A LEGAL FILE NAME.

**

FMGR 054

DISC NOT MOUNTED
AN ATTEMPT WAS MADE TO REFERENCE AN UNMOUNTED DISC CARTRIDGE. MOUNT THE DISC CARTRIDGE USING THE 'MC' COMMAND. IF UNDER SESSION CONTROL THE 'AC' COMMAND COULD BE USED INSTEAD TO ALLOCATE DISC SPACE WITH THE SPECIFIED CRN.

**

FMGR 055

MISSING PARAMETER
A REQUIRED PARAMETER HAS BEEN OMITTED. CHECK THE COMMAND AND RE-ENTER IT WITH THE MISSING PARAMETER.

**

FMGR 056

BAD PARAMETER
A PARAMETER WAS SPECIFIED INCORRECTLY OR A TRACK PARAMETER SPECIFIES A TRACK THAT IS OUTSIDE THE RANGE OF THE FMGR TRACKS. CHECK THE COMMAND AND RE-ENTER IT CORRECTLY.

**

FMGR 057

BAD TRACK NOT IN FILE AREA
THE SPECIFIED TRACK IS IN THE SYSTEM AREA OR IS A DIRECTORY TRACK. CORRECT THE COMMAND AND RE-ENTER IT.

**

FMGR 058

LG AREA EMPTY
AN ATTEMPT WAS MADE TO SAVE THE CONTENTS OF THE LG AREA WHICH IS EMPTY. RECOMPILE THE PROGRAM OR USE THE 'MR' COMMAND.

**

FMGR 059

REPORTED TRACK UNAVAILABLE
A RE-INITIALIZATION ATTEMPT LOWERED THE FIRST TRACK INTO THE SYSTEM AREA. THE LAST TRACK IS REPORTED. RE-ENTER THE COMMAND WITH THE FIRST TRACK SPECIFIED AS THE LAST TRACK + 8 (THE MINIMUM).

**

FMGR 060

DO YOU REALLY WANT TO PURGE THIS DISC?
A RE-INITIALIZATION ATTEMPT RAISES THE FIRST TRACK OR LOWERS THE DIRECTORY TRACKS INTO THE FILE AREA AND WILL DESTROY A FILE. ENTER '??' OR 'NO' TO STOP THE REINITIALIZATION. ENTER 'YES' TO CONTINUE.

**

FMGR 061

DO A "DC" AND A "MC" ON THIS CR
AN ATTEMPT WAS MADE TO REPLACE A MOUNTED CARTRIDGE WITH AN CARTRIDGE THAT HAS NOT BEEN PREVIOUSLY INITIALIZED WITHOUT ENTERING A 'DC' AND A 'MC' COMMAND. ENTER A 'DC' AND 'MC' COMMAND FOR THIS CARTRIDGE.

**

FMGR 062

MORE THAN 63 DISCS
AN ATTEMPT WAS MADE TO MOUNT THE 64TH CARTRIDGE (THE LIMIT IS 63 CARTRIDGES). DISMOUNT A CARTRIDGE TO MAKE ROOM, IF POSSIBLE.

**

FMGR 063

EXCEEDING SESSION DISC LIMIT
AN ATTEMPT IS BEING MADE TO MOUNT MORE DISCS TO A SESSION THAN IS ALLOWED IN THE USER ACCOUNT FILE. DISMOUNT AN UNUSED DISC AND RE-ENTER THE COMMAND.

**

FMGR 064

NO DISC AVAILABLE FROM DISC POOL
ALL DISCS IN DISC POOL ARE ALLOCATED OR NO DISCS BIG ENOUGH ARE AVAILABLE.

**

FMGR 065

CONFLICT IN SST DEFINITION

**

FMGR 066

NO ROOM IN SST

**

FMGR 067

PROGRAM NOT FOUND

NOTE

Table F-2 shows the standard 7-bit set-code positional order and notation with bit 7 the high-order and bit 1 the low-order bit position.

For example, the code for R is: b7b6b5b4b3b2b1
1 0 1 0 0 1 0

Table F-2. ASCII/Binary Conversion

b7					0	0	0	0	1	1	1	1
b6					0	0	1	1	0	0	1	1
b5					0	1	0	1	0	1	0	1
BITS b4	b3	b2	b1	COLUMN → / ROW ↓	0	1	2	3	4	5	6	7
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

APPENDIX G

GLOSSARY OF TERMS

Absolute System	- The absolute binary code of the Real Time Executive System (stored in logical unit 2).
Auxiliary Disc	- The disc is optional and when used, is assigned to logical unit 3. (The absolute binary code of RTE does not reside on the auxiliary disc). When the auxiliary disc is part of a moving head disc drive, it is contained on one subchannel of that drive. The auxiliary disc has the same status in the RTE as does the system disc in that it is treated as a logic extension of the system disc.
Controller	- The interface unit between the computer and the Disc Drives, which controls Disc Drive operation.
Device Down	- Relates to the state of a peripheral device. When the device is down, it is no longer operable. Also, it refers to the operator command DN, which sets the devices down.
Device Up	- Relates to the state of a peripheral device. When the device is up it is operable. Also, it refers to the operator command UP, which sets the device up after it has been set down.
Disc Drive	- Consists of a mechanism to rotate the disc, and electronic circuitry to write data on and read data off the disc through disc heads.
Moving Head Disc Drive	- Consists of a mechanism to rotate magnetic discs. There is one head per recording surface that is attached to a movable arm. The head is moved to the addressed track by means of an actuator driving the arm and head.
Peripheral Disc	- A peripheral disc is a disc that is available to the user for read/write operations but for which the Real Time Executive (RTE) does not manage the disc nor maintain a track assignment table. A peripheral disc must have a logical unit number assignment greater than 6.
Program Swapping	- When program A is removed from computer memory and stored on the disc in its current state of execution, and program B is placed in the computer

memory area (for execution) formerly occupied by program A. Program A is eventually returned to memory and continued.

RTE Executive - The total operating system comprised of three program modules, EXEC, SCHED, and RTIOC, plus I/O drivers, and various tables.

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

Scratch Area - A number of disc tracks used during system generation for storage of the relocatable binary code of RTE.

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

Time Out - Relating to the state of a peripheral device. When the device has time out, it is no longer operable. It is also the amount of time the RTE will wait for the device to respond to an I/O transfer command before RTE makes the device inoperable.