PREFACE

This manual contains operating and programming information for the Distributed System Interface Kit (HP 91701A/B). The reader is assumed to be familiar with the material presented in the following Hewlett-Packard publications:

A Pocket Guide to the 2100 Computer

Real-Time Executive Software System Manual

Real-Time Executive File Manager Manual

Basic Control System Manual

The HP FORTRAN Programmer's Reference Manual

The HP FORTRAN IV Programmer's Reference Manual

The HP ALGOL Programmer's Reference Manual

The RTE System Cross Loader Manual

12771A Computer Serial Interface Kit Operating and Service Manual

12773A Computer Modem Interface Kit Operating and Service Manual

TABLE OF CONTENTS

APPENDICES

LIST OF ILLUSTRATIONS

LIST OF TABLES

GLOSSARY OF TERMS USED IN THIS MANUAL

BCS TERMINAL – A terminal whose software includes the Basic Control System. Input/Output is accomplished by BCS compatible drivers. I/O may be buffered or non-buffered.

CENTRAL COMPUTER – The computer which resides at the "hub" of a spoke-configured distributed system. Generally thought of as having the most extensive set of peripherals or resources.

COMMUNICATION LINE – The interconnecting link between communications cards in computers of a distributed system. The link is most commonly a cable comprised of a double set of twisted wire pairs with shielding.

COMMUNICATION SERIAL INTERFACE DRIVER – A communication input/output subroutine which passes preformatted requests and data to and from the serial interface cards connecting computers. See also "Computer Communication."

COMPUTER COMMUNICATION – The transmission of coded requests and associated blocks of data between computers. The transmission is over communication lines and utilizes communication serial interface I/O cards. The coded requests are typically 20 computer words in length and contain all information required by the receiving computer to reconstitute and execute the request. The data blocks are of varying length.

DISTRIBUTED SYSTEM – Generally used to mean a multi-computer environment. HP 9701 is a Distributed System in the sense that multiple physical terminal computers are tied to a central (host) computer.

DRIVER – Software <input>/<output> routine which performs actual I/O functions on devices. Always written in assembly level language.

HOST SYSTEM – Central computer in which the HP 91701 central software resides.

LOADER – This term is used in two ways in this manual:

1. A program which transforms, or "LOADS," relocatable object programs into absolute object programs resolving symbolic addresses, external references, segmentation links, and setting indirect addresses for instruction operands that do not reside on the same page as the instruction. Examples are:

   SXL – A loader which will generate programs that can be executed in an environment different from that in which it was produced.

   RTE Loader – A background disc-resident program that:

   a. Relocates background programs for RTE.
   b. Adds or replaces RTE disc-resident programs.

2. A program which communicates with a storage device for the purpose of transferring another program from the device into execution memory. The storage device may be local or remote. In the remote case the action is referred to as a down-link load.

MASTER/SLAVE RELATIONSHIP – Master/slave relationship dictates the task relationship between computers in a distributed system. The computer which typically issues remote requests (the terminals in a HP 9701 Distributed System) is called the master. The program which executes the requests (the HP 9701 central computer) is called the slave.

MODEM – An electronic device for converting a binary bit pattern to a modulated sine wave for transmission across a telephone line and then back to a bit pattern at the demodulation end.

MODULE – A unit of software designed to provide a specified response to a given system stimulus. Loosely, a "program." The total HP 9701 Distributed System software is broken up into modules to facilitate design, production and modification to meet varying installation requirements.

MONITOR – In the HP 91701 context, monitor is used to mean that program module (or modules) which are used to process the requests of the associated stream. The remote file access monitor (RFAM) is an example.

MULTIPROCESSOR INTERLEAVING – Interleaving is a special process of addressing adjacent storage modules in and even/odd fashion.

PROGRAM-TO-PROGRAM COMMUNICATION – The communication between two software modules executing in two different computers. HP 91701 supports the initialization of the program at the central computer from the terminal computer. The format of the requests and data passed between the programs is defined by the user who interfaces directly with the communication serial interface driver. See "Computer Communication."

PROTECTED AREA OF CORE – The last 64 (decimal) words of 2100 memory are hardware read/write protected. This area contains the communication bootstrap (TCE/1), or the HP Basic Binary Loader (BBL), or the HP Basic Binary Disc Loader (BBDL). See Appendix F.

REMOTE EXEC CALLS – Remote exec calls consist of a

subset of the standard RTE Exec calls performed by a central (host) computer on behalf of a user program requesting service from a remote terminal computer.

REMOTE FILE ACCESS – Remote file access consists of file actions performed by a central (host) computer File Management Package (FMP) on behalf of a user program requesting from a remote terminal computer.

REMOTE PROGRAM LOAD – A three (optionally four) step communication process, remote program loading consists of:

1. A request for a specific program file (stored on mass storage at central) to be loaded into the requesting remote terminal.
2. The access of that file by central station software and outputting absolute format file records.
3. The receiving of those records and their storage by the terminal computer.
4. Optional execution of the received code by transfer to a passed starting address.

ROUND ROBIN DISPATCH – A method of servicing incoming requests which assures equal service to all terminal computers. HP 91701 assures no single terminal computer is allowed two consecutive services while there is a request outstanding on an unserviced computer.

SERIAL INTERFACE – Hard wired interface between two computers. (See "Modem.")

SIO - SUBROUTINES – A group of System Input/Output (SIO) subroutines used to perform basic input/output operations for programs in absolute form.

STREAM TYPE – A group of logically related requests comprise a stream type and are processed by one program monitor. The various aspects of remote file access (remote open, remote close, etc.) comprise the remote file stream.

SUBSTREAM – An identifier assigned to a request by the terminal computer and maintained until status is returned to the terminal user following service.

TABLE DRIVEN MODULE – A design philosophy widely employed in the HP 9701 Distributed System. Table driven modules consist of a relatively simple process module which derives information for a specific action from one or more tables of information and related subroutines.

TERMINAL COMPUTER – A computer (perhaps one of several) connected by a communication line to a (processing center) host computer. The terminal computer is usually dedicated to a specific task and shares central computer resources with other terminals.

TERMINAL OPERATING SYSTEM – Related software modules loaded into a terminal computer in a distributed system which permit:

1. The communication between the terminal computer and the central station computer, and,
2. The communication between the terminal computer and the terminal computer operator.

TERMINAL REQUEST QUEUE – A pool of input/output buffers which contain the incoming requests from time of receipt until the request information is overlayed with completion status information for transmission back to the requesting terminal.

TRANSMISSION LOG – A parameter returned following an I/O transfer to indicate the number of words or characters actually transmitted.

TRANSMISSION PARAMETER BUFFER – The formatted system request passed from computer to computer to facilitate communication. (See "Computer Communication.")

SECTION I
GENERAL DESCRIPTION

INTRODUCTION

This manual describes the HP 91701A/B Distributed Systems Interface Kits as they are used in the HP 9701 Distributed System. Tables 1-1 and 1-2 list the components of the HP 91701A/B Kits.

The HP 9701 Distributed System is composed of a network of minicomputer-based terminals connected and integrated into a central real-time, multi-programming data processing system. The central system in this configuration is the HP 9600E/F Real-Time Executive (RTE) System. The terminal systems consist of any member of the HP 2100 Series of computers operating in the Basic Control System (BCS) environment. The Distributed System capability results from the HP 91701 Distributed Systems Interface Kit. The component interfaces of the HP 9701 Distributed System are shown in Figure 1-1, below.

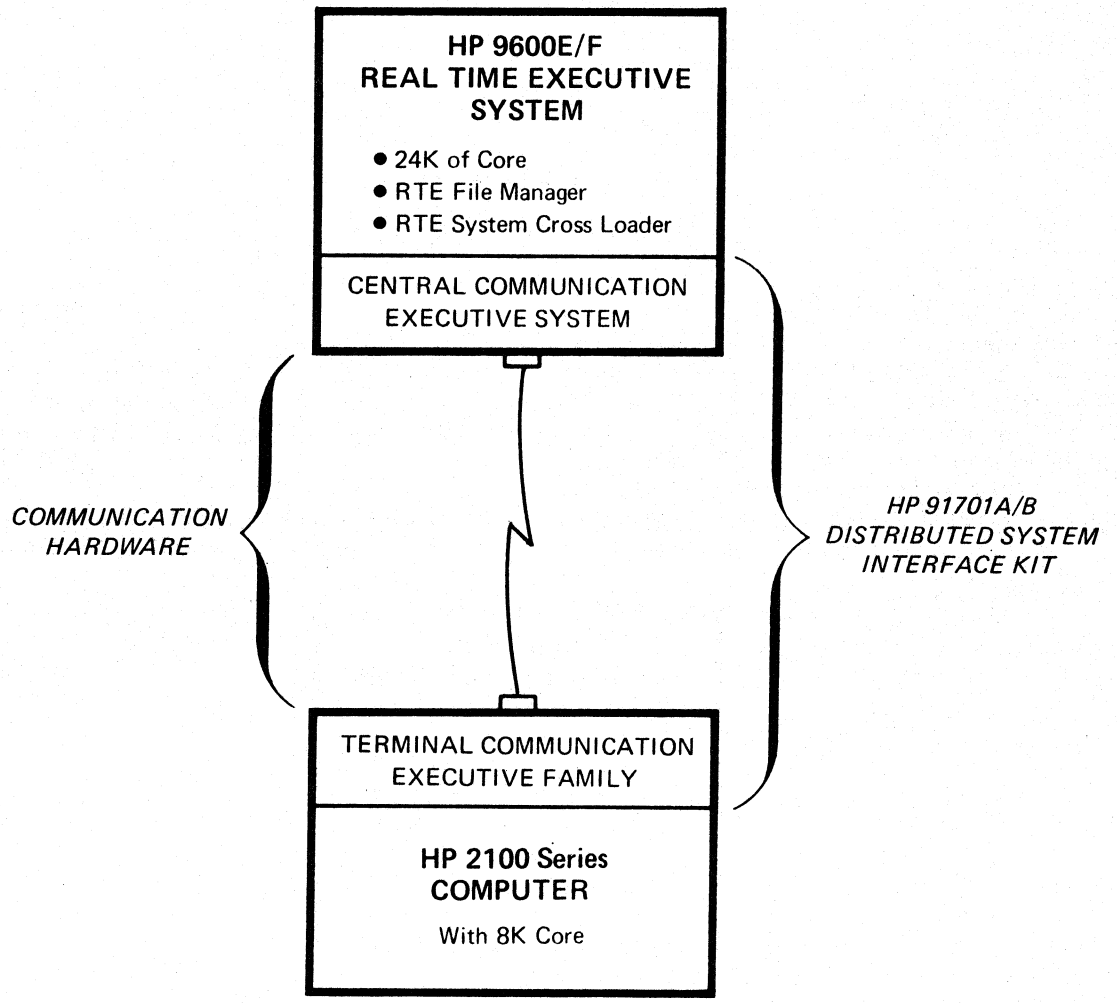


Figure 1-1. HP 9701 Distributed System Minimum Requirements

HP 9701 REQUIREMENTS

The HP 9701 Distributed System requires that the HP 9600E/F RTE at the central have a 24K core size and that the RTE File Manager is part of the user's system. The RTE System Cross Loader (SXL) assists in the central program development for the remote terminal systems.

For minimal system design, the HP 91701 Distributed System Interface Kit requires as little as 8K core at the remote terminal system.

SYSTEM COMPONENTS

The HP 91701 Distributed System Interface Kit facilitates the integration of the remote terminal systems and the central RTE into a multi-computer distributed system. The HP 91701 Distributed System Interface Kit is available in two models, as shown in Tables 1-1 and 1-2. These models are distinguished by the types of communication hardware provided.

The HP 91701A (Table 1-1) offers the user a hardwire interface between the two computers with the HP 12771A Serial Communication Interface Kit.

The HP 91701B (Table 1-2) offers the user communication between two computers via common carrier telephone lines using user supplied data sets and the HP 12773A Modem Interface Kit.

Table 1-1. 91701A Distributed System Interface Kit

Hardware	Software	Documentation	Options	Additional Requirements
HP 12771A Computer Serial Interface Kit	Distributed System RTE Software Kit Distributed System BCS Software Kit	12771A Computer Serial Interface Kit Operating and Service Manual 91701 Distributed System Kit Manual	020 041 or 042 or 043 or 044 or 045 or 046	9600E RTE System or 9600F RTE System and HP 2100 Series Computer (BCS)

Options:

020 Distributed System Kit (RTE/BCS) for an additional terminal system (after the first) includes hardware for hardwired communications at both ends and an additional terminal communications executive.

041 250-foot communication cable with loose connectors for HP 91701A or HP 91701A-020.

042 250-foot communication cable with assembled connectors for HP 91701A or HP 91701A-020.

043 500-foot communication cable with loose connectors for HP 91701A or HP 91701A-020.

044 500-foot communication cable with assembled connectors for HP 91701A or HP 91701A-020.

045 975-foot communication cable with loose connectors for HP 91701A or HP 91701A-020.

046 975-foot communication cable with assembled connectors for HP 91701A or HP 91701A-020.

Table 1-2. 91701B Distributed System Interface Kit

Hardware	Software	Documentation	Options	Additional Requirements
12773A 2 each Computer Modem Interface Cards	Distributed System RTE Software Kit Distributed System BCS Software Kit	12773A Computer Modem Interface Kit Operating and Service Manual 91701 Distributed System Kit Manual	020 023	9600E RTE System or 9600F RTE System and HP 2100 Series Computer (BCS) Customer Furnished Modems

Options:

020 Distributed System Kit (RTE/BCS) for an additional terminal includes hardware to support modem communications at both ends and an additional terminal communication executive.

023 Distributed System Kit (BCS) for an additional terminal, with hardware to support modem communication only at the terminal system. Requires previous HP 91701B or HP 91701B-020.

The HP 91701 Distributed System Interface Kit software is described in Table 1-3 below. This list does not include the software for either the HP 9600E/F RTE central, or the HP 9600A BCS terminals.

It should be noted that the RTE Executive, binary tape number 29016-60001, provided with Real-Time Executive Software, must be replaced at system generation with the HP 91701 Distributed System version of this module, binary tape number 29100-60011.

Table 1-3. HP 91701A/B Distributed System Interface Kit Software

Binary Tape Number	Description	Binary Tape Number	Description	Binary Tape Number	Description
RTE CENTRAL		BCS TERMINAL		SIO DRIVERS	
29100-60002	QUEUE	29037-60001	TCE/1 (4K)	29100-60016	Buffered TTY Driver (24K)
29100-60003	DISP	29037-60002	TCE/2 (4K)	29100-60017	TTY Driver LP-Compatible (24K)
29100-60004	TAM (RTE)	29037-60003	TEXEC		
29100-60005	RFAM	29037-60004	TAM (BCS)	29100-60018	System Dump (24K)
29100-60006	DISC	29037-60005	RFAIN	29100-60019	Photo Reader Driver (24K)
29100-60007	PROGL	29037-60006	.IOC.	29100-60020	Tape Punch Driver (24K)
29100-60008	DLIST	29037-60007	D.ØØD	29100-60021	HP 2778A Line Printer Driver (24K)
29100-60009	ERR	29037-60008	TCE/1 (8K)		
29100-60010	LSTEN	29037-60009	TCE/2 (8K)	29100-60022	HP 2767 Line Printer Driver (24K)
29100-60011	EXEC	29037-60010	TCE/1 (12K)		
29100-60012	QUDIS (8 terminals)	29037-60011	TCE/2 (12K)	29100-60023	HP 7970 Magnetic Tape Driver (24K)
29100-60013	QUDIS (16 terminals)	29037-60012	TCE/1 (16K)		
29100-60014	QUDIS (32 terminals)	29037-60013	TCE/2 (16K)	29100-60024	HP 2020 Magnetic Tape Driver (24K)
29100-60015	QUDIS (24 terminals)	29037-60014	TCE/1 (24K)		
		29037-60015	TCE/2 (24K)	29100-60025	HP 3030 Magnetic Tape Driver (24K)
		29037-60016	TCE/1 (32K)		
		29037-60017	TCE/2 (32K)		

29100 - 60032 TC3GN
29103 - 60001 SXL
29103 -60119 BCSGN

SYSTEM DESCRIPTION

MULTI-COMPUTER ENVIRONMENT

The computer network provided by the 9701 system provides the advantage of geographical distribution and the economy of shared peripherals. The multi-computer environment of the 9701A/B also permits parallel processing between the central computer and all terminals.

MULTIPROGRAMMING

The multi-computer network of the 9701 Distributed System maximizes the multiprogramming capability of RTE. The RTE central with its ability to overlap and inter-leave terminal data can make maximum use of shared resources.

CENTRAL PROGRAM PREPARATION

Application programs may be completely prepared at the central computer. This can be done while the central computer is handling communications and processing for other terminal systems. Application programs are written, compiled, configured and relocated on the central disc, ready for immediate load and execution in the terminal system.

REMOTE PROGRAM LOAD

The terminal systems may load programs from the central disc with a single request. This provides a minimum terminal system with a high speed disc capability for program storage. It provides the ability to use standard test programs at several terminal systems and to have them controlled and maintained centrally. Remote program load also provides the capability for terminal systems with small core to handle large programs which have been segmented and chained.

REMOTE FILE ACCESS

Each terminal system can read or write data to the central disc through the use of Remote File Access commands. This provides a small terminal system with the complete resources of the File Management Package software, central computer mass storage, including high speed disc, and central computer peripheral access.

PROGRAM-TO-PROGRAM COMMUNICATION

Program-to-program communication consists of a dialogue between a program at the central computer and a program at a terminal computer. A communications link is established between the two programs via the BCS Communications Driver (D.65), at the terminal and the RTE Communications Driver (DVR65) at the central computer.

REMOTE I/O

The terminal programmer and operator have access to all logical units defined at the central computer as if they existed at the local terminal. These units normally include disc, teletype, high-speed punch, tape reader, line printer, and magnetic tape.

SECTION II

SOFTWARE MODULE DEFINITIONS

The Distributed System software modules are divided by location into those modules which reside at the terminal computers and those modules which reside at the central computer. The particular modules required in a given terminal configuration are determined by the level of Terminal Communications Executive, (TCE/1, TCE/2 or TCE/3), installed.

This section discusses each of the Terminal Communications Executives separately and outlines the modules required by each. The modules at the central computer are also indicated with discussion of their purpose.

PART I – TCE/1 MODULE DEFINITION

The TCE/1 Overview in Figure 2-1 shows the interface points between the user, the Distributed System software and the central computer. The TCE/1 executive is a modified version of the Basic Binary Loader. It requires no other subroutines. Appendix E provides an octal listing of TCE/1. Section III contains information on running application programs under this terminal executive.

PART II – TCE/2 MODULE DEFINITION

The TCE/2 Overview shown in Figure 2-2 is intended to show the points of interface between the user, the Distributed System terminal software and the central computer software. TCE/2 is a program that requires no other subroutines. Section III contains a discussion of how to run application programs under this executive. Section IV provides information on the operator requests, error codes and initialization procedures for this program.

PART III – TCE/3 MODULE DEFINITIONS

The TCE/3 Overview in Figure 2-3 indicates the interface between the application operator/programmer, the terminal software and the central computer. TCE/3 consists of relocatable BCS modules configured to provide both a self-contained loader and a terminal control program.

The available program calls, operator requests and error messages are provided in Section IV. This section also provides the initialization procedure. Section III contains information on application program code and formats requirements for this program.

PART IV – CENTRAL MODULE DEFINITIONS

The Central Overview in Figure 2-4 shows the interface between 91701 software, the 12771A software, and the

9600E/F software. This also indicates the module interface between the terminal user and the central processing software. What may not be apparent in this figure is that the central user retains the full capability of the 9600 software, (RTE and FMP packages), while adding the 91701 capability.

Section III contains a summary of the application program requirements. Section IV contains summaries of the program calls, operator requests and error messages for both RTE and FMP. If further information is required, the reader is referred to the appropriate system software manual.

PART I

TCE/1 MODULE DEFINITION

TCE/1 is a special 64-word Basic Communications Bootstrap Loader which is used in a way similar to the standard Basic Binary Loader. TCE/1 resides in the protected area of core in place of the standard BBL and is available to restart the terminal in the event of a software crash. TCE/1 can be used to load programs selected by octal number 1 - 77 input through the switch register. The 91701 software at the central computer prefixes PROG to the switch register number and loads program PROG *nn* from the central disc into the terminal computer. The program PROG *nn* must have been previously stored on disc as absolute code. This terminal executive can be used for initial startup and for terminal systems with minimum peripherals. It supports terminal computers with 4K or more of core.

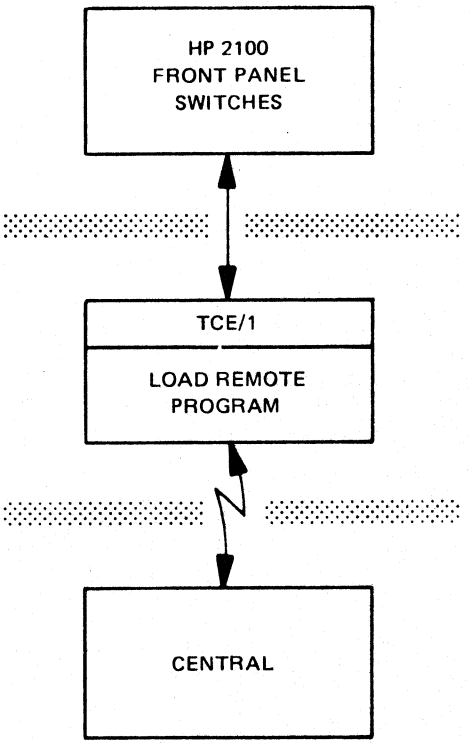


Figure 2-1. TCE/1 Terminal Overview

PART II

TCE/2 MODULE DEFINITION

TCE/2 is a small, simple, non-interrupt executive. It is a stand-alone, self-contained module which resides in high core. While TCE/2 is not a member of the SIO family, it will operate in either a user-supplied SIO, or BCS environment as described in Section III.

This version of the terminal executive combines the ability to load programs from the central computer with the ability to execute them via commands entered through the keyboard. Multiple LOAD commands can be used to test the effects of overlays and patches. The command parameters are made available to the loaded program by passing the address of the parameter list in the B-Register. I/O is internal for communications and CRT/TTY. It supports terminal computers with 4K or more of core. The use of TCE/2 is limited to program load and execution at the terminal computer.

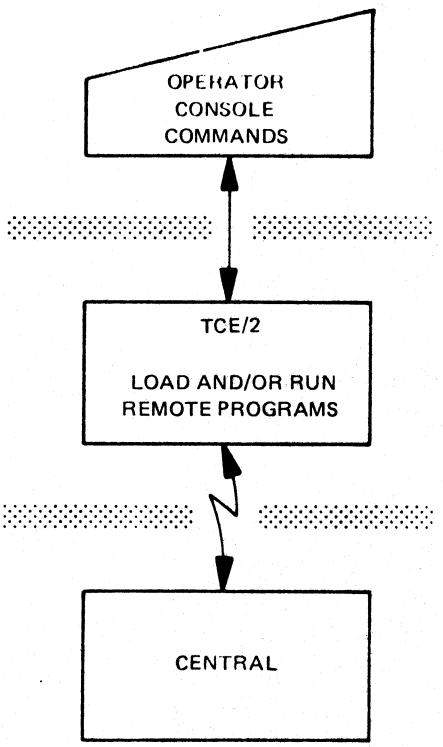


Figure 2-2. TCE/2 Terminal Overview

PART III

TCE/3 MODULE DEFINITIONS

The TCE/3 terminal executive consists of the modules shown in Figure 2-3. These relocatable BCS modules provide the full 91701 capability to the terminal user. TCE/3 is fully upward compatible with TCE/2. In addition to the functions available with TCE/2, the TCE/3 user may do remote file accesses and program-to-program communication.

TCE/3 consists of the following modules:

D.ØØD	CRT/TTY Driver
TEXEC	Terminal Executive Module
RFAIN	Remote File Access Initiator
#TAM	Terminal Access Monitor
D.65	BCS Communications Driver
.IOC.	BCS I/O Control Subroutine

CRT/TTY DRIVER (D.ØØD)

A modified version of the standard D.ØØD–TTY driver is used to perform special functions for TCE/3. It is always enabled for input (even during output). All input interrupts are checked to see if the ESCAPE key was pressed. If the ESCAPE key is pressed, the terminal program is aborted. If any key is pressed during output, I/O stops and the driver "listens" for the next input. If the key pressed was the ESCAPE key, the terminal program is aborted. If the key pressed is anything but ESCAPE, I/O resumes.

RUBOUT (line deletion) and Control H (backspace) are processed during operator input. Input of a carriage return causes completion, and a line feed is output by the driver.

TERMINAL EXECUTIVE (TEXEC)

TEXEC, the interface between the terminal operator and the TCE/3 remote file access modules, provides remote operations through operator requests. This executive accepts all TCE/2 commands and in addition allows: scheduling of programs to be run at the central computer, messages to be sent to the central operator, and a request to be made for the time from central's real-time clock. Disc files at the central computer may be created, renamed, closed and purged.

Terminal operator requests are checked for errors and converted into the corresponding RFAIN calling sequence. TEXEC then performs the call to RFAIN (Remote File Access Initiator). The allowable commands are organized into a mnemonic table, where each ASCII name corresponds to the address of a processing routine.

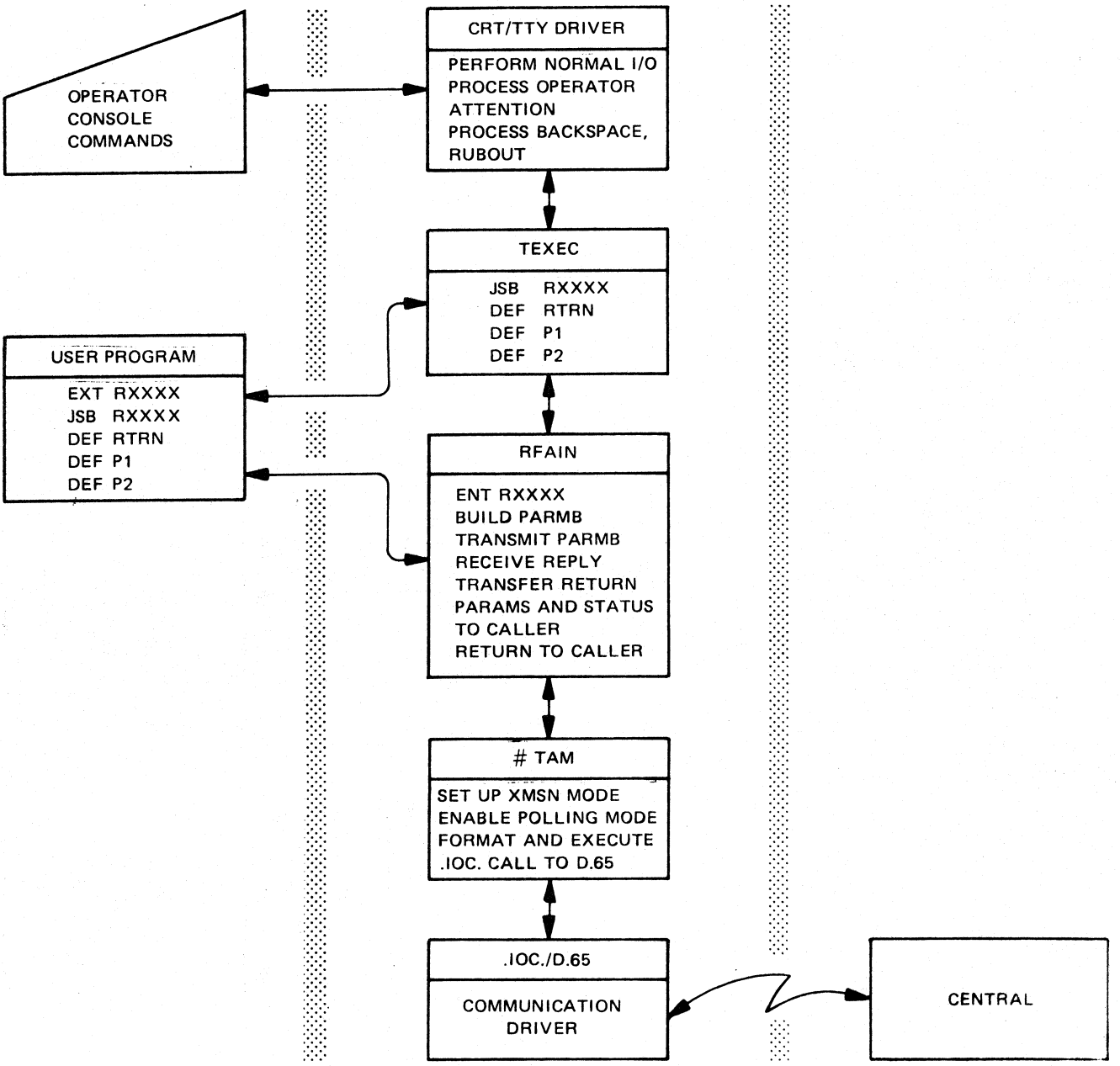


Figure 2-3. TCE/3 Overview

REMOTE FILE ACCESS INITIATOR (RFAIN)

RFAIN contains entry points for remote RTE File Management Package calls, and remote RTE Executive calls. Upon entry into this module, the user's parameter addresses are placed in the Parameter Address Vector table (PAV). An empty entry in the Request Array is obtained and the addresses of the user's return parameters are stored. The index number of the entry becomes the sub-stream number for the call. The Request Array is single entry for BCS terminals.

Information for each remote request is placed in the parameter buffer (PARMB) to be transmitted to the central computer. The PARMB contains: a data stream ID which specifies the class of calls, a subsystem ID which is used as an index to request array entry for multiprogramming terminals, and a function code that corresponds to the particular user and the parameter values required by central to reconstruct this call are placed into PARMB. A call is made to #TAM (Terminal Access Monitor), to transmit PARMB to the central computer. The status of the remote communications and the status of the remote call are returned via #TAM into the associated request array entry's input buffer. When the remote call is completed, the return parameters are transferred to the user and RFAIN returns control to its caller.

TERMINAL ACCESS MONITOR (#TAM)

#TAM is the interface to the BCS D.65 driver, described in Appendix A. It converts user calls to .IOC. formats.

HP 12771A COMPUTER SERIAL INTERFACE KIT DRIVER (D.65)

D.65 is a driver used in the BCS terminal environment to set up calling sequences to .IOC., and to communicate with the RTE communications driver DVR65 at the central computer. D.65 with DVR65 provides the data link between the central computer and all terminal computers. The following calls are legal for D.65:

- Receive or transmit data only
- Receive request only
- Enable Listen mode
- Enable Polling mode
- Transmit STOP reply
- Clear request
- Status request

Appendix A provides a description of driver characteristics, call sequences data transmission mode requirements, and programming techniques.

PART IV

CENTRAL MODULE DEFINITIONS

The primary function of the 91701 software at the central computer is to provide remote data and program storage, and retrieval for the terminal computers. The modules which constitute the 91701 central software are shown in Figure 2-4. The central computer operator and programmer are provided with all of the capability normally found in the RTE and FMP packages. In addition to these functions, the 91701 software provides the ability to do program-to-program communication. A program at central may communicate with a program running at a terminal after the central program initiates the data link and communication.

Figure 2-4 shows an overview of the central computer software and its interface with the terminals. This figure also indicates the relationship that exists in the 9701 system between the 91701 central software and the RTE and FMP subsystems.

The 91701 software at the central computer is composed of the following software modules:

LSTEN	Enable Listen Mode
DVR65	RTE Communications Driver
QUEUE	System Queue Module
DISP	System Request Dispatches
TAM	Telecommunications Access Monitor
RFAM	Remote File Access Monitor
DISC	Disc Service Subroutine
QUDIS	Privileged Queueing/Dequeueing Subroutine
ERR	Error Monitor
PROGL	Program Load
DLIST	Directory List

RTE and FMP are also required and function in the 9701 system as indicated in the discussion in the paragraphs which follow.

ENABLE LISTEN MODE MODULE (LSTEN)

This module is used to configure the 91701 software and establish the communications link between the central computer and the terminals. LSTEN has two options which are used to establish communication links for the entire system or to reinitialize the communication link between central and a single terminal. LSTEN is turned on by operator request from the central computer console.

RTE COMMUNICATIONS DRIVER (DVR65)

DVR65 is a driver used in the RTE central environment to communicate with the BCS environment at the terminals

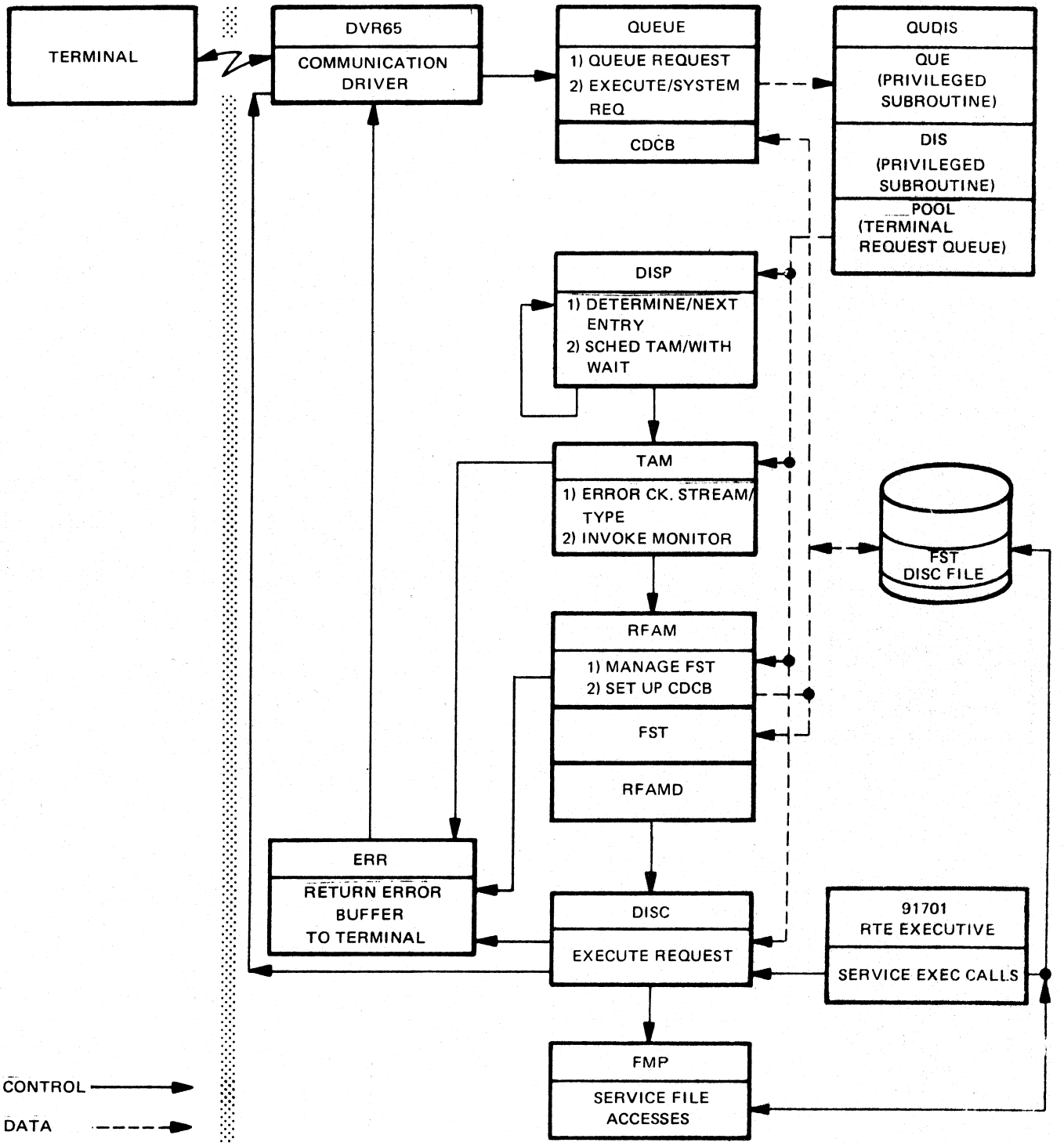


Figure 2-4. Central Overview

through D.65 to recognize the interrupt from a terminal request.

Appendix A describes the driver characteristics, call sequences, data transmission mode requirements and programming techniques executed at the terminal.

SYSTEM QUEUE MODULE (QUEUE)

The module QUEUE shown in Figure 2-5 determines the Logical Unit number of the requesting terminal. It receives all terminal requests for all stream types and places them into a service queue. The queueing scheme is as follows:

a. Terminals are allowed one service in turn (round-robin fashion).

b. Hardware priority is maintained. The round-robin begins at the highest priority terminal on each round.

QUEUE is designed in such a way that the mixture of communication executive types is transparent to the system.

QUEUE receives the standard PARMB buffers from TCE/2 or TCE/3, or builds a 20-word PARMB buffer from the one program number input by TCE/1.

QUEUE also decodes and services special system requests that do not require monitor intervention.

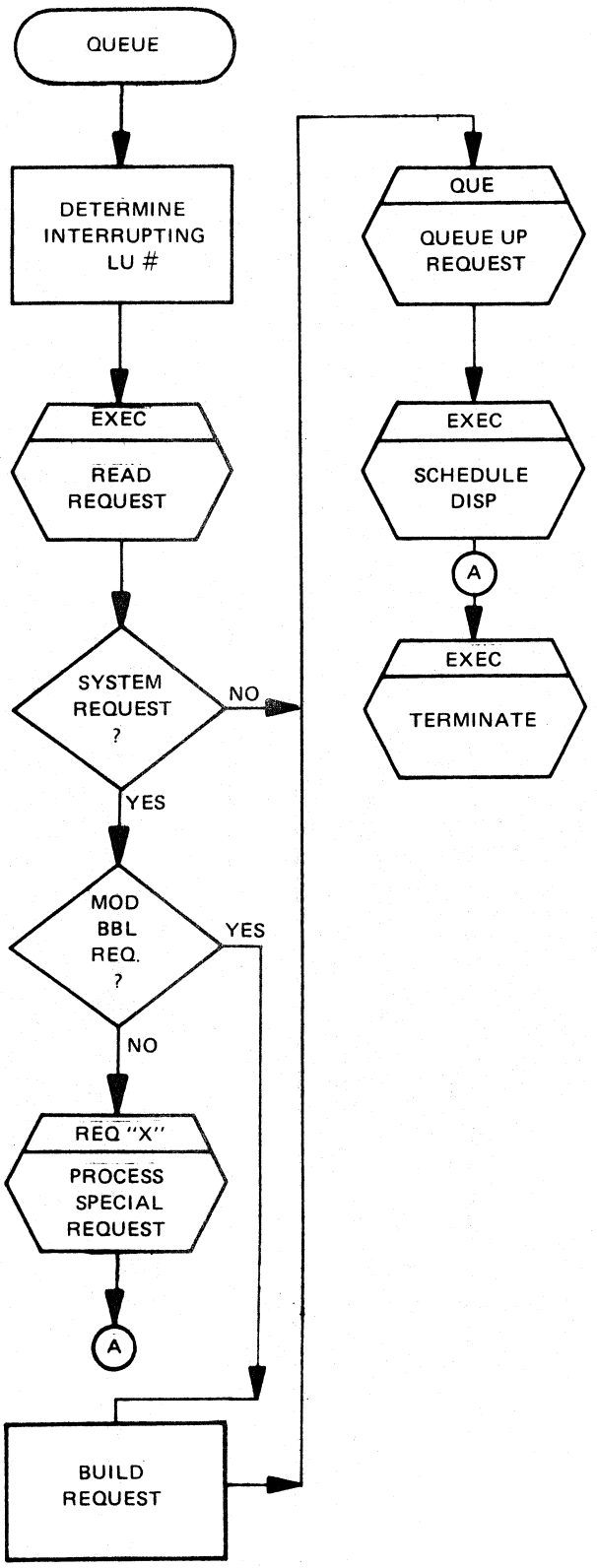


Figure 2-5. Queue Detail

SYSTEM REQUEST DISPATCHER MODULE (DISP)

The module DISP shown below in Figure 2.6 is scheduled at the conclusion of each terminal service operation and at the conclusion of any queueing action. DISP has two functions:

1. Determine the next request to be serviced according to the round robin priority established in QUEUE,

2. Initiate service for the request by scheduling the Telecommunications Access Monitor (TAM).

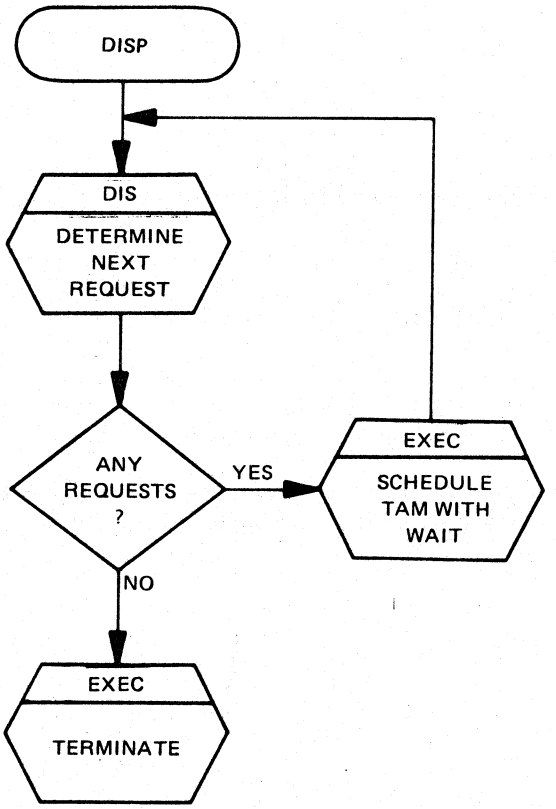


Figure 2-6. DISP Detail

TELECOMMUNICATIONS ACCESS MONITOR (TAM)

The module TAM is scheduled by DISP. It receives a pointer to the request buffer to be serviced. The request buffer is examined and the stream type extracted from the PARMB header. The stream type is utilized as a switch to invoke the monitor required to process the request.

The Distributed System interface (91701) provides a Remote File Access Monitor (RFAM) and an Error Monitor (ERR). Legal Remote File Access stream types cause RFAM to be scheduled. Illegal stream types cause ERR to be scheduled.

REMOTE FILE ACCESS MONITOR (RFAM)

The Remote File Access Monitor is invoked by TAM for Remote File Access and remote EXEC calls.

RFAM, as shown in Figure 2-7, maintains a File State Table (FST) with entries and work areas for each remote file open in the system. The attempt is made to maintain as many FST entries in core as possible to decrease access time. Those FST entries that cannot be maintained in core (oldest entries) are put in the overflow file and maintained on disc. The location of these entries is made available by the Remote File Access Monitor Directory, which is also maintained by RFAM.

After a file access is cleared through the FST, a processing subroutine (DISC) is called to perform the disc requests.

DISC SERVICE SUBROUTINE (DISC)

DISC, as shown in Figure 2-8, is a table-driven module which is called to examine the transmission parameter buffer (PARMB) and reconstitute it into a File Access calling format so that a local FMP or RTE call can be made for processing call. This is accomplished by extracting the original calling sequence parameters from the PARMB, and using the sub-function number from PARMB as an index into a table of entry points for the appropriate processing subroutine to be called.

Each reconstituted file command is preceded and followed by its pre-process and post-process subroutine. A typical pre-processing subroutine can perform preliminary error checking or data buffer gathering directly from the serial interface driver, DVR65. A typical post-processing subroutine can return status or data on the request directly to the terminal of origin.

ERROR MONITOR (ERR)

The error monitor is scheduled by TAM when a stream type error is detected. It is also scheduled by RFAM or DISC when a completion error is detected in either of these modules. ERR receives the error type that has been detected in the B-Register. ERR returns this information in a completion request to the originating terminal to signal that the request has failed. After transmission of the request, ERR returns control to the module that scheduled it and the servicing of the original request is concluded.

PRIVILEGED QUEUEING/DEQUEUEING SUBROUTINE (QUDIS)

QUDIS consists of two subroutines, each with a separate entry point. The first, QUE, is called from QUEUE; the second, DIS, is called from DISP. QUE receives a request read from the interface and places it in the Terminal Request Queue (POOL) buffer pool for later servicing. QUE also sets an activity flag to indicate a request has been re-

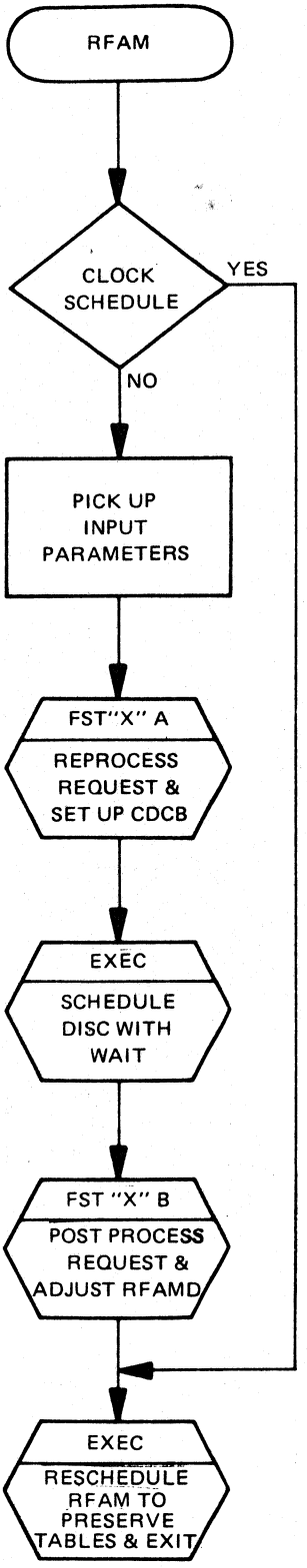


Figure 2-7. RFAM Detail

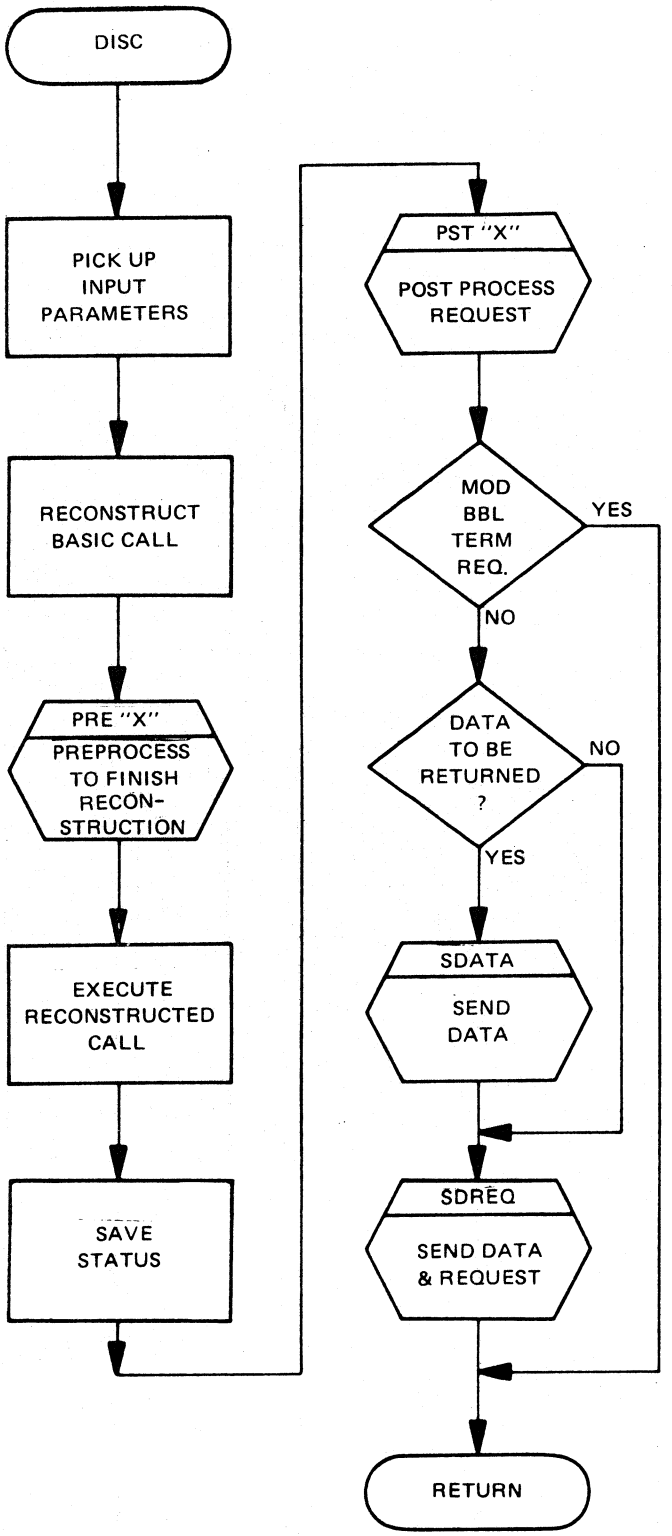


Figure 2-8. DISC Detail

ceived and awaits service. DIS tests all activity flags and determines which one of the outstanding requests are to be returned to DISP for service. DIS clears the activity flag on the request about to be serviced and maintains the round-robin dispatching scheme.

REAL-TIME EXECUTIVE (RTE) – Special RTE EXEC Module Provided With 91701

The central computer programmer and operator have access to the full capability of RTE as described in the **Real-Time Executive System Manual** and as summarized in Section IV of this manual. The remote user has access through the 91701 software to those central computer RTE calls and requests which are meaningful at the terminal. See Section IV for a complete description of the functions available. It should be remembered that the RTE executive module (EXEC) furnished with the 91701 Distributed System Interface Kit must replace the standard RTE executive at system generation.

REAL-TIME EXECUTIVE FILE MANAGER SYSTEM (FMP)

The central computer programmer and operator have access to the full capability of FMP as detailed in the **Real-Time Executive File Manager System Manual** and summarized in Section IV of this manual. The Distributed System Interface software package (91701) allows the remote user access to all of the program callable functions implemented by FMP. The 91701 calling sequences for FMP, however, are modified to include remote status information.

Data buffer length for remote read and write has been limited to 128-word blocks, to simplify buffer management. This constraint can be avoided by the user if a Remote Exec call is invoked to schedule a user program. The data transfer is then accomplished by program-to-program communication.

PROGRAM LOAD (PROGL)

PROGL is provided to handle the terminal user's remote program load request for a complete or partial absolute file load from the central computer. PROGL is scheduled under RTE from the terminal by a Remote Exec call. The name of the program to be loaded is passed to PROGL as an input parameter. The Logical Unit number of the requesting terminal at central is passed to PROGL.

DIRECTORY LIST (DLIST)

DLIST permits the terminal operator to request a partial or complete listing of files currently being maintained by FMP. An optional 6-character filter word, entered by the terminal operator, is used to determine which file information is to be provided for partial listings. An optional cartridge label may also be entered. If the cartridge label is included, only that cartridge is listed.

The HP 91701 interface kit software will fully support a virtually unlimited file list convention through the use of the character filter. When the filter word is entered, only those files whose names contain characters that match the filter characters are output. If the character filter is used, six characters must be entered. Character positions to be ignored are specified by an asterisk (*).

SECTION III

APPLICATION PROGRAM PREPARATION

This section is divided into four parts that describe the procedures, formatting conventions and functions of the TCE/1, TCE/2, TCE/3 executives, and central.

PART I – TCE/1 APPLICATION PROGRAM PREPARATION

This part describes the format for application programs required by TCE/1. It also discusses the configuration of a System Input/Output (SIO) environment and a Basic Control System (BCS) environment in which application programs can be run under TCE/1. Information is provided on how to store application programs on the central disc and how to load them into the terminal computer.

PART II – TCE/2 APPLICATION PROGRAM PREPARATION

Application programs to be run under the TCE/2 executive may be run in an Absolute System Input/Output (SIO) environment, or in a relocatable Basic Control System environment. This part describes the formats for programs to be run in either environment. It describes how to load programs on the central disc and how to bring them down to the terminal to be run.

PART III – TCE/3 APPLICATION PROGRAM PREPARATION

This part describes the application program formats required by the TCE/3 executive. It also describes how to store application programs on the central computer disc and how to load them into a terminal computer to be run under TCE/3. A sample program to list the contents of remote source files is provided on Page 3-8.

PART IV – CENTRAL APPLICATION PROGRAM PREPARATION

Application programs run at the central computer must be in the formats required by the Real-Time Executive System (RTE) and the Real-Time Executive File Manager System (FMP). Execution procedures remain the same as described in the **Real-Time Executive Software System Manual** and in the **Real-Time Executive File Manager Programming and Operating Manual.**

PART I

TCE/1 APPLICATION PROGRAM PREPARATION

TCE/1 operates in either an SIO or a BCS environment. If application programs are to be run in an SIO environment, the user should consult the **HP Software Operating Procedures Manual** and the **HP Small Programs Manual.** Additional information on the BCS system may be found in the **HP Basic Control System Manual.**

TCE/1 IN AN SIO ENVIRONMENT

The following requirements must be implemented for TCE/1 to be operable in an SIO environment:

- Application programs must be assembled using the absolute option of the assembler.
- The required SIO drivers must be assembled into a single module with the correct I/O channels and ORG statements for the configuration being used.
- Application program modules are stored on the central computer disc with the FMP store request:

  ST,5,PROG*nn*,BA

  where:

  LU number 5 is the standard input device.

  PROG*nn* is the TCE/1 required format for the program name. The variable, *nn*, may be any number between $\emptyset - 77_8$.

  BA indicates that the required record format is binary absolute.

- SIO drivers are stored on the central disc in a file separate from the application programs. The file name for the SIO drivers must also be in the PROG*nn* format.

  ST,5,PROG*nn*,BA

  The parameters of the STore request for SIO drivers are the same as the parameters used to store the application programs.

- TCE/1 is loaded into terminal core as described in Part I of Section IV.

The resulting terminal core configuration is shown in Figure 3-1.

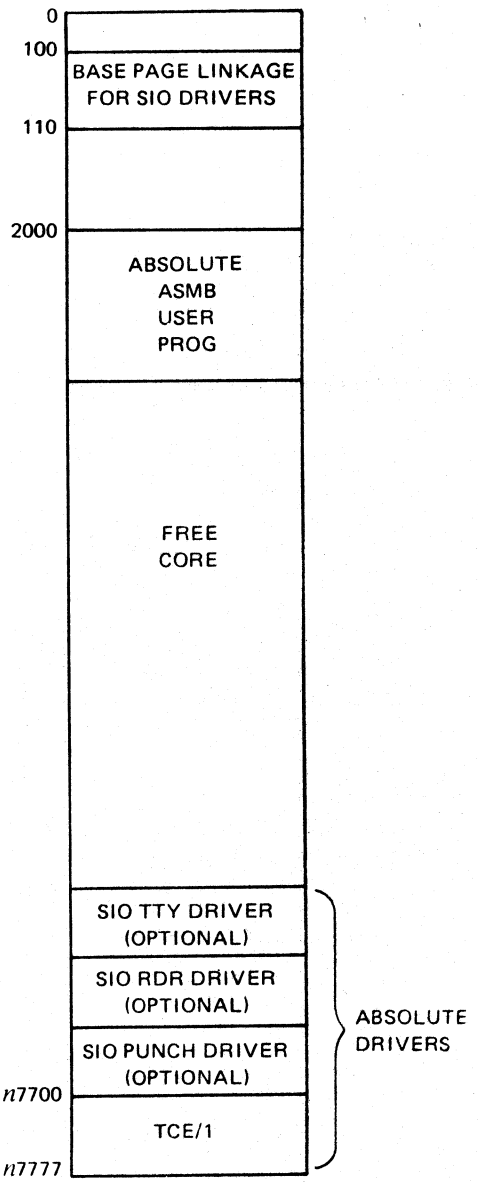


Figure 3-1. TCE/1 in an SIO Environment

TCE/1 IN A BCS ENVIRONMENT

The BCS environment may be configured in one of two ways: by using the SXL Loader, or through the standard PCS. An example of a TCE/1 – BCS environment produced by the SXL Loader is shown in Figure 3-2. An example of a PCS-produced BCS environment is shown in Figure 3-3.

SXL Loader BCS Configuration

Three kinds of modules are required for this configuration:

TCE/1

Application programs

BCS drivers and relocatable library described in the **Basic Control System Manual.**

The modules are loaded onto the central disc using the SXL Loader and brought from the disc into the terminal computer using TCE/1. The sequence of steps used to accomplish this are detailed below.

Load TCE/1 into the terminal computer (if not already in).

Assemble relocatable application programs.

Use the SXL Loader to load application programs and BCS modules onto the central disc.

ON,SXL
- OUTPUT ABS ON PROG*nn*
etc.

The output statement must specify absolute format. The file name under which the application program and BCS modules are stored must be in the format PROG*nn*. See SXL Loader manual for a complete description of commands and the TCE/3 material in this section for a sample command file.

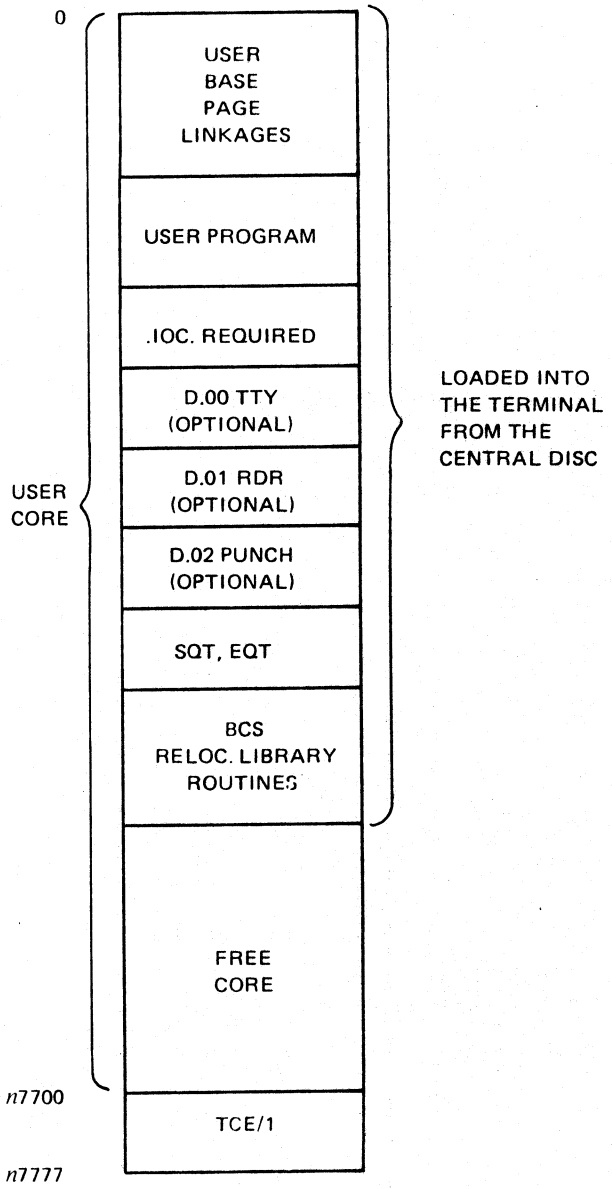


Figure 3-2. TCE/1 in an SXL-Produced BCS Environment

PCS-PRODUCED BCS CONFIGURATION

The required BCS drivers, the BCS Loader and .IOC. are input to the Prepare Control System program described in the **Basic Control System Manual.** The output from this process is an absolute BCS Loader tape which is used to configure the application program.

The BCS Loader tape is input to the computer followed by the application program, assembled in relocatable format. The BCS library is searched to resolve undefined externals and an absolute tape is output. This tape contains the application program linked to the IOC environment with library references resolved.

The absolute user program which results from the PCS process is loaded on the central disc with the File Management Package STore command:

:ST,5,PROG*nn*,*BA*

where:

LU number 5 is the standard input device.

PROG*nn* is the required format for the file name.

BA is the required Binary Absolute record format.

Central files are loaded into the terminal computer by TCE/1 as described in Section IV.

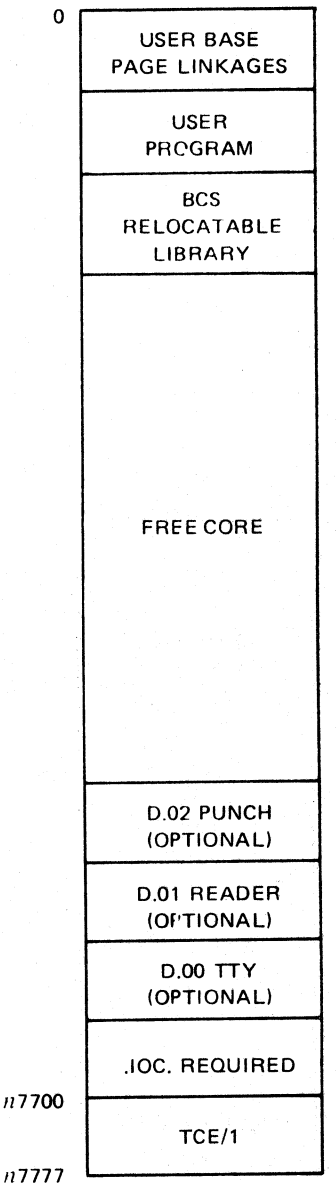


Figure 3-3. TCE/1 in a PCS-Produced BCS Environment

PART II

TCE/2 APPLICATION PROGRAM PREPARATION

TCE/2 operates in either an SIO or a BCS environment. If application programs are to be run in an SIO environment, the user should consult the **HP Software Operating Procedures Manual** and the **HP Small Programs Manual.** Additional information on the BCS system may be found in the **HP Basic Control System Manual.**

TCE/2 IN AN SIO ENVIRONMENT

The following requirements must be implemented for TCE/2 to be operable in an SIO environment.

- The required SIO drivers must be reassembled with the correct I/O channels and ORG statements for the configuration being used. The memory required for the last driver cannot extend beyond location *n*6400$_8$.

  where *n* = Ø – 4K, 1 – 8K, 2 – 12K, 3 – 16K, 5 – 24K, 7 – 32K

- SIO drivers are stored on the central disc with the FMP STore request:

  ST,5,*file name*, BA

  where:

  LU number 5 is the standard input device.

  file name is a name for the file which meets the requirements of the TCE/2 LOAD request described in Section IV of this manual. The file name need not be in the PROG*nn* format required by TCE/1.

  BA specifies Binary Absolute record format.

- The parameters of the STore request for application programs are the same as the parameters used to store the SIO drivers.

- TCE/2 is loaded into terminal core as described in Part II of Section IV.

- Application programs are brought into terminal core from the central disc with the TCE/2 LOAD request.

  LOAD,*file name* (application program)
  LOAD,*file name* (SIO drivers)

  The SIO driver file is loaded last so that driver linkages are assured.

  The resulting terminal core configuration is shown in Figure 3-4.

- Begin execution of application programs by pressing RUN.

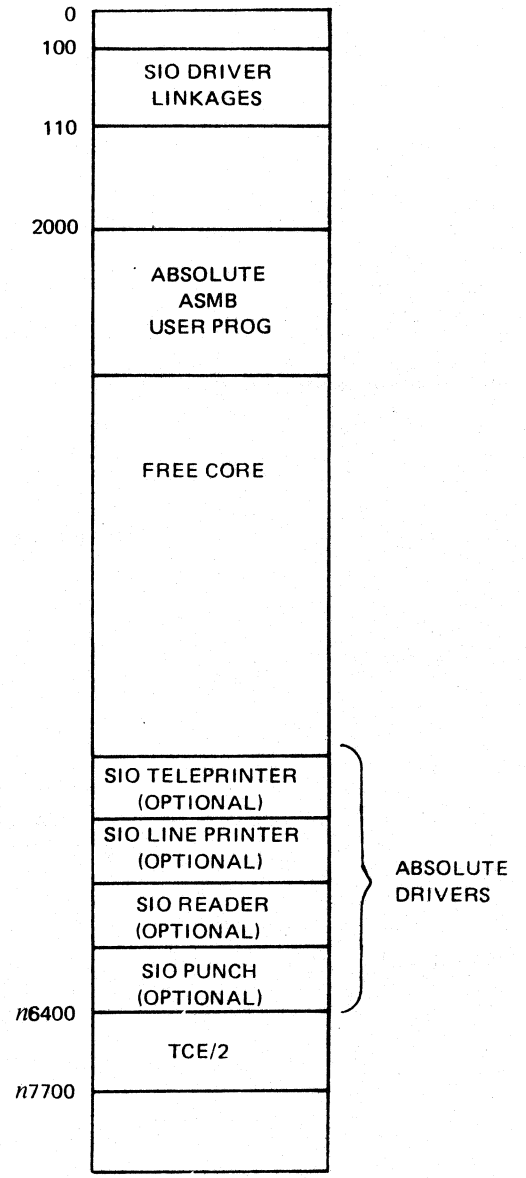


Figure 3-4. TCE/2 in an SIO Environment

TCE/2 in a BCS Environment

The BCS environment for TCE/2 can be configured using either the SXL Loader or the standard PCS. A core map sample for TCE/2 in a BCS environment created by the SXL Loader is shown in Figure 3-5. TCE/2 in a BCS environment produced by PCS is shown in Figure 3-6.

SXL LOADER BCS CONFIGURATION

The process of using the SXL Loader to generate a TCE/2 environment in a BCS configuration is described below. The three types of modules involved are TCE/2, application programs, and the BCS modules.

- TCE/2 must be loaded into terminal core as described in Section VI of this manual.
- Application programs are assembled in relocatable format.
- The preparation of BCS drivers, the BCS Loader, and relocatable library are described in the **Basic Control System Manual.**
- When all of the modules are prepared, they are loaded onto the central disc using the SXL Loader and brought from the disc into the terminal computer by TCE/2. A complete example of the SXL command file is provided in Part III of this section. The TCE/2 requirements for the command file are described below.

  ON,SXL,1
  – OUTPUT ABS ON *program name*
  – BOUNDS LWAM = *n*6377, FWAM = 2ØØØ
  – RELOCATE
  etc.

The output statement must specify absolute format, the file name under which the application program and BCS modules are stored must be compatible with the TCE/2 LOAD and RUN requests. The BOUNDS statement must specify the starting location, the ending location, and command area to prevent TCE/2 from being overlayed. The RTE System Cross Loader manual should be consulted for a complete description of commands.

Files stored on the central disc are brought into the terminal computer for execution with the TCE/2 RUN request. The SXL Loader automatically sets the starting address of the program at location 2.

The resulting terminal core configuration is shown in Figure 3-5.

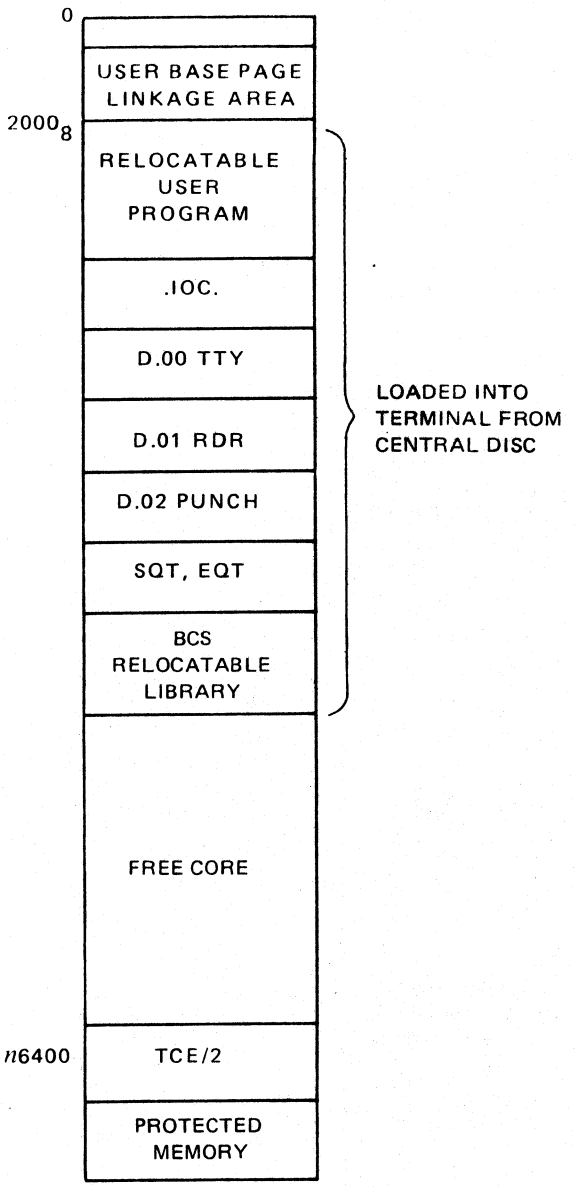


Figure 3-5. TCE/2 in an SXL-Produced BCS Environment

PCS-PRODUCED BCS CONFIGURATION

The required BCS drivers, the BCS Loader and .IOC. are input to the Prepare Control System program described in the **Basic Control System Manual.** The output from this process is an absolute BCS Loader tape which is used to configure the application program.

The BCS Loader tape is input to the computer followed by the application program, assembled in relocatable format. The BCS library is searched to resolve undefined external and an absolute tape is output. This tape contains the application program linked to the IOC environment with library references resolved.

The absolute user program which results from the PCS process is loaded on the central disc with the File Management Package STore command:

:ST,5, *file name,* BA

where:

LU number 5 is the standard input device.

file name is the user defined program name.

BA is the required Binary Absolute record format.

Central files are loaded into the terminal computer by TCE/2 using the LOAD request.

The resulting terminal core configuration is shown in Figure 3-6.

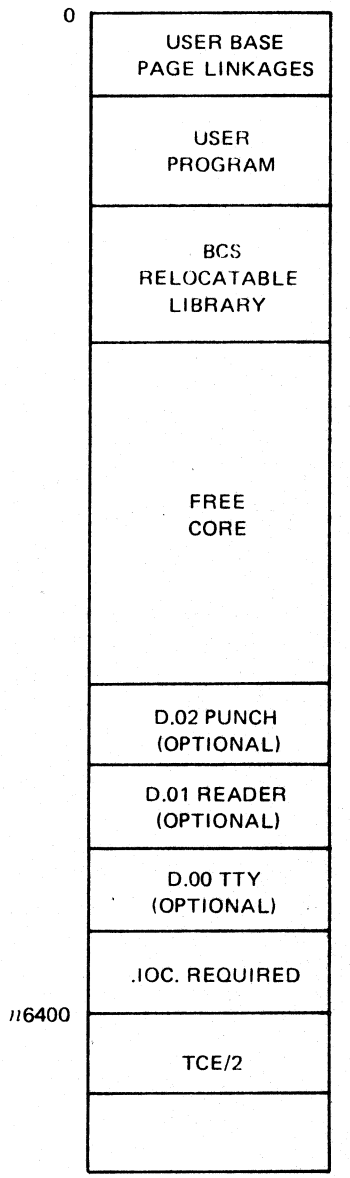


Figure 3-6. TCE/2 in a PCS-Produced Environment

PART III

TCE/3 APPLICATION PROGRAM PREPARATION

The relocatable modules of TCE/3 contain BCS drivers and .IOC. as described in Section VI so that a BCS operating environment is assumed for TCE/3. A complete description of the BCS system is contained in the **HP Basic Control System Manual.**

Application programs are loaded into the computer in relocatable formats. The SXL Loader is used to link the application programs, the TCE/3 executive and any library requirements into absolute format which can be run at the terminal. A description of SXL Loader requests used for this purpose is provided below, followed by a sample program to list the contents of remote source files.

ON,SXL,1	Commands input to the loader are coming from the TTY.
– ECHO ON 6	Both input commands and resulting output are listed on the line printer.
– OUTPUT ABS ON LISTER	Output file in absolute format.
– MAP MODULES, GLOBALS	Program names and entry points will both be listed.
– TR, TCESNP	Transfers control to the loader command file set up by the snap statement at the end of the TCE/3 generation (See Section VI). The TCESNP file sets up entry points to the TCE/3 resident.
– RELOCATE 9D	Relocates the application program (on the tape reader in this example) from the tape reader to core.
– SEARCH F4MTR	Search libraries to satisfy additional FORTRAN requirements (source code for this example is HP FORTRAN IV).
– SEARCH BCSL IB	Search BCS library to satisfy remaining external references.
– DISPLAY UNDEFS	Display undefineds
– END	

The application program can be executed at the terminal with the following request:

```
RUN LISTER
```

Listed below is the source code for a sample program to list remote source files resident on the central disc at the terminal CRT. It is this program for which the Loader commands above are used.

```
FTN4,B,L,A
      PROGRAM LISTF
      DIMENSION IBUF(40),NAME(3)
      WRITE(2,101)
 101  FORMAT(" UTILITY PROGRAM TO LIST REMOTE SOURCE"/
     1" FILES ON TERMINAL TTY.")
      WRITE(2,102)
 102  FORMAT(" CENTRAL DISC FILE NAME?")
      READ(1,103) (NAME(I),I=1,3)
 103  FORMAT(3A2)
      CALL ROPEN(ISTAT,IERR,NAME)
      IF((ISTAT.EQ.-1).AND.(IERR.GE.0)) GO TO 5
      WRITE(2,104)
 104  FORMAT(" REMOTE FILE OPEN ERROR: ABORT")
      GO TO 60
   5  DO 10 I=1,40
  10  IBUF(I) = 20040B
      CALL RREAD(ISTAT,IERR,NAME,IBUF,40,L)
```

```
      IF((ISTAT.EQ.-1).AND.(IERR.GE.0)) GO TO 20
      WRITE(2,105)
105   FORMAT(" REMOTE FILE READ ERROR: ABORT")
      GO TO 50
 20   IF (L.EQ.-1) GO TO 50
      IF (L.LE.34) GO TO 30
      L=34
 30   WRITE(2,106) (IBUF(I),I=1,L)
106   FORMAT(" ",34A2)
      CALL ESCON(IV)
      IF (IV) 50,5,50
 50   CALL RCLOS(ISTAT,IERR,NAME)
      IF((ISTAT.EQ.-1).AND.(IERR.GE.0)) GO TO 60
      WRITE(2,107)
107   FORMAT(" REMOTE FILE CLOSE ERROR: ABORT")
 60   END
      END$
```

SXL LOADER MAP

```
****** T=00000 IS ON LU 09

0001    -OUTPUT ABS ON LISTER
0002    -MAP MODULES, GLOBALS ON 4
0003     FILE   PROGRAM GLOBAL  FIRST    LAST       BASE PAGE    REF.BY
0004     NAME   MODULE  VARS.   ADDRS.   ADDRS.
0005     ------------------------------------------------------------
0006    -TR,TCESNP
0007    --BOUNDS FWAM =012151,LWAM=017677,FWABP= 000474,LWABP= 001777,&
0008    --FWAC=000000,LWAC=000000
0009    -- SET LOCC     TO  012151
0010                    LOCC     012151
0011    -- SET BPLOCC   TO  000474
0012                    BPLOCC   000474
0013    -- SET #TAM    TO 006433
0014                    #TAM     006433
0015    -- LINKS START AT  000230, 006433
0016    -- SET D.65    TO 010000
0017                    D.65     010000
0018    -- SET ATTEN   TO 002103
0019                    ATTEN    002103
0020    -- LINKS START AT  000334, 002103
0021    -- SET RLOAD   TO 003224
0022                    RLOAD    003224
0023    -- SET CHAIN   TO 003606
0024                    CHAIN    003606
0025    -- SET RNPGM   TO 003512
0026                    RNPGM    003512
0027    -- SET RMESG   TO 003524
0028                    RMESG    003524
0029    -- SET IDLE    TO 002001
0030                    IDLE     002001
```

```
0031    -- SET PTPON  TO 003674
0032                   PTPON   003674
0033    -- SET PTPOF  TO 003701
0034                   PTPOF   003701
0035    -- SET ESCON  TO 003706
0036                   ESCON   003706
0037    -- SET HALT   TO 002001
0038                   HALT    002001
0039    -- SET EXEC   TO 002001
0040                   EXEC    002001
0041    -- SET STOP   TO 002001
0042                   STOP    002001
0043    -- SET .ENTR  TO 005431
0044                   .ENTR   005431
0045    -- LINKS START AT  000322, 005431
0046    -- SET $BUSY  TO 006030
0047                   $BUSY   006030
0048    -- LINKS START AT  000120, 006030
0049    -- SET RFATM  TO 004561
0050                   RFATM   004561
0051    -- LINKS START AT  000205, 004561
0052    -- SET GETLU  TO 004576
0053                   GETLU   004576
0054    -- LINKS START AT  000227, 004576
0055    -- SET RCRET  TO 004641
0056                   RCRET   004641
0057    -- LINKS START AT  000223, 004641
0058    -- SET RPURG  TO 004670
0059                   RPURG   004670
0060    -- LINKS START AT  000226, 004670
0061    -- SET ROPEN  TO 004702
0062                   ROPEN   004702
0063    -- SET RREAD  TO 004715
0064                   RREAD   004715
0065    -- SET RWRIT  TO 004734
0066                   RWRIT   004734
0067    -- SET RPOSN  TO 005005
0068                   RPOSN   005005
0069    -- SET RWIND  TO 005017
0070                   RWIND   005017
0071    -- SET RCLOS  TO 005027
0072                   RCLOS   005027
0073    -- LINKS START AT  000225, 005027
0074    -- SET RNAME  TO 005040
0075                   RNAME   005040
0076    -- LINKS START AT  000224, 005040
0077    -- SET RCONT  TO 005054
0078                   RCONT   005054
0079    -- SET RLOCF  TO 005066
0080                   RLOCF   005066
0081    -- SET RAPOS  TO 005076
0082                   RAPOS   005076
0083    -- SET RSTAT  TO 005111
0084                   RSTAT   005111
0085    -- SET REXEC  TO 005131
0086                   REXEC   005131
```

```
0087    -- LINKS START AT  000207, 005131
0088    -- SET TMOUT  TO 007042
0089                    TMOUT   007042
0090    -- SET INT65  TO 007020
0091                    INT65   007020
0092    -- LINKS START AT  000440, 007020
0093    -- SET .IOC.  TO 007074
0094                    .IOC.   007074
0095    -- LINKS START AT  000122, 007074
0096    -- SET DMAC1  TO 007311
0097                    DMAC1   007311
0098    -- LINKS START AT  000435, 007311
0099    -- SET DMAC2  TO 007312
0100                    DMAC2   007312
0101    -- LINKS START AT  000436, 007312
0102    -- SET IOERR  TO 007270
0103                    IOERR   007270
0104    -- LINKS START AT  000437, 007270
0105    -- SET XSQT   TO 007307
0106                    XSQT    007307
0107    -- SET XEQT   TO 007310
0108                    XEQT    007310
0109    -- LINKS START AT  000257, 007310
0110    -- SET .BUFR  TO 007242
0111                    .BUFR   007242
0112    -- SET D.00   TO 007313
0113                    D.00    007313
0114    -- SET I.00   TO 007445
0115                    I.00    007445
0116    -- SET $ESC   TO 007736
0117                    $ESC    007736
0118    -- LINKS START AT  000121, 007736
0119    -- SET I.65   TO 011004
0120                    I.65    011004
0121    -- SET .SQT.  TO 012121
0122                    .SQT.   012121
0123    -- SET .EQT.  TO 012127
0124                    .EQT.   012127
0125    -- LINKS START AT  000255, 012127
0126    -- SET .MEM.  TO 012140
0127                    .MEM.   012140
0128    -- LINKS START AT  000105, 012140
0129    -RELOCATE 9D
0130            LISTF           012151  012711  000474  000503
0131                    LISTF   012151
0132    -SEARCH F4MTR
0133            F4MTR           012712  012762  000504  000505
0134                    .RIO.   012712
0135                    .IIO.   012721                          LISTF
0136                    .RAY.   012730
0137                    .IAY.   012743
0138    -SEARCH LIBD
0139            F4MTR           012763  015036  000506  001241
0140                    .DIO.   014440                          LISTF
0141                    .BIO.   014513
```

```
0142                    .IOI.    014335
0143                    .IOR.    014310
0144                    .IAR.    014374                                  F4MTR
0145                    .RAR.    014350                                  F4MTR
0146                    .DTA.    014536                                  LISTF
0147          MPY                014037  015147  001242  001241
0148                    .MPY     014037                                  FRMTR
0149          FLOAT              015150  015154  001242  001241
0150                    FLOAT    015150                                  FRMTR
0151          .PACK              015155  015261  001242  001241
0152                    .PACK    015155                                  FRMTR
0153          DLDST              015262  015317  001242  001241
0154                    .DLD     015262                                  FRMTR
0155                    .DST     015272                                  FRMTR
0156          IFIX               015320  015354  001242  001241
0157                    IFIX     015320                                  FRMTR
0158          .FLUN              015355  015367  001242  001241
0159                    .FLUN    015355                                  FRMTR
0160          CLRIO              015370  015374  001242  001241
0161                    CLRIO    015370                                  LISTF
0162   -DISPLAY UNDEFS
0163   -END
0164   NO UNDEFS
0165   TRANSFER TO  LISTF    012151
0166   CORE USED    012151   015374  000474  001241
0167   FREE CORE    015375   017677  001242  001777
0168   OUTPUT FILE NAME:LISTER
0169  $END SXL
```

PART IV

CENTRAL APPLICATION PROGRAM PREPARATION

Complete descriptions of application program consideration used at the central computer are provided in the **Real-Time Executive Software System Manual** and the **Real-Time Executive File Manager Manual.**

The process of preparing an application program at the central computer to be run at either the central computer or at a terminal computer is shown in Figure 3-7. The central programming aids which are available as RTE options include the RTE Editor, the RTE FORTRAN compilers, the ALGOL Compiler, the RTE Assembler, the relocatable libraries, and the RTE System Cross Loader (SXL).

The RTE Editor creates, lists and edits source language tapes and disc files. The Editor requires two inputs: an input file containing edit commands, and a symbolic file containing source programs. A complete description of the RTE Editor can be found in the **Real-Time Executive Software System Programming and Operating Manual.**

The FORTRAN compilers accept source modules from either an input device or a source file created by the RTE Editor. The compilers translate the source programs into relocatable object code which can be punched on tape or stored on the load-and-go tracks of the central disc. FORTRAN II is described in the **HP FORTRAN Programmer's Reference Manual.** HP FORTRAN IV is described in the **HP FORTRAN IV Programmer's Reference Manual.** The use of the FORTRAN Compilers in the RTE System is described in the **Real-Time Executive Software System Manual.**

The ALGOL Compiler accepts source modules from either an input device or a source file created by the RTE Editor. The compiler translates the code into relocatable object format and punches it on tape or stores it on the load-and-go tracks of the central disc. A complete description of the ALGOL Compiler is contained in the **HP ALGOL Programmer's Reference Manual.** ALGOL Compiler operation in an RTE environment is described in the RTE Manual.

The RTE Assembler accepts source modules from either an input device or a source file created by the RTE Editor. It translates the source programs into either absolute or relocatable object code. The relocatable code is punched on paper tape or stored on the load-and-go tracks of the central disc, or both. The RTE Assembler is described in the **Assembler Programmer's Reference Manual.** The use of the Assembler is described in the RTE Manual.

The Relocatable Libraries used at the central computer are the standard RTE relocatable libraries described in the RTE Manual. The relocatable libraries used at the terminals are the Basic Control System relocatable libraries described in the **HP Basic Control System Manual.**

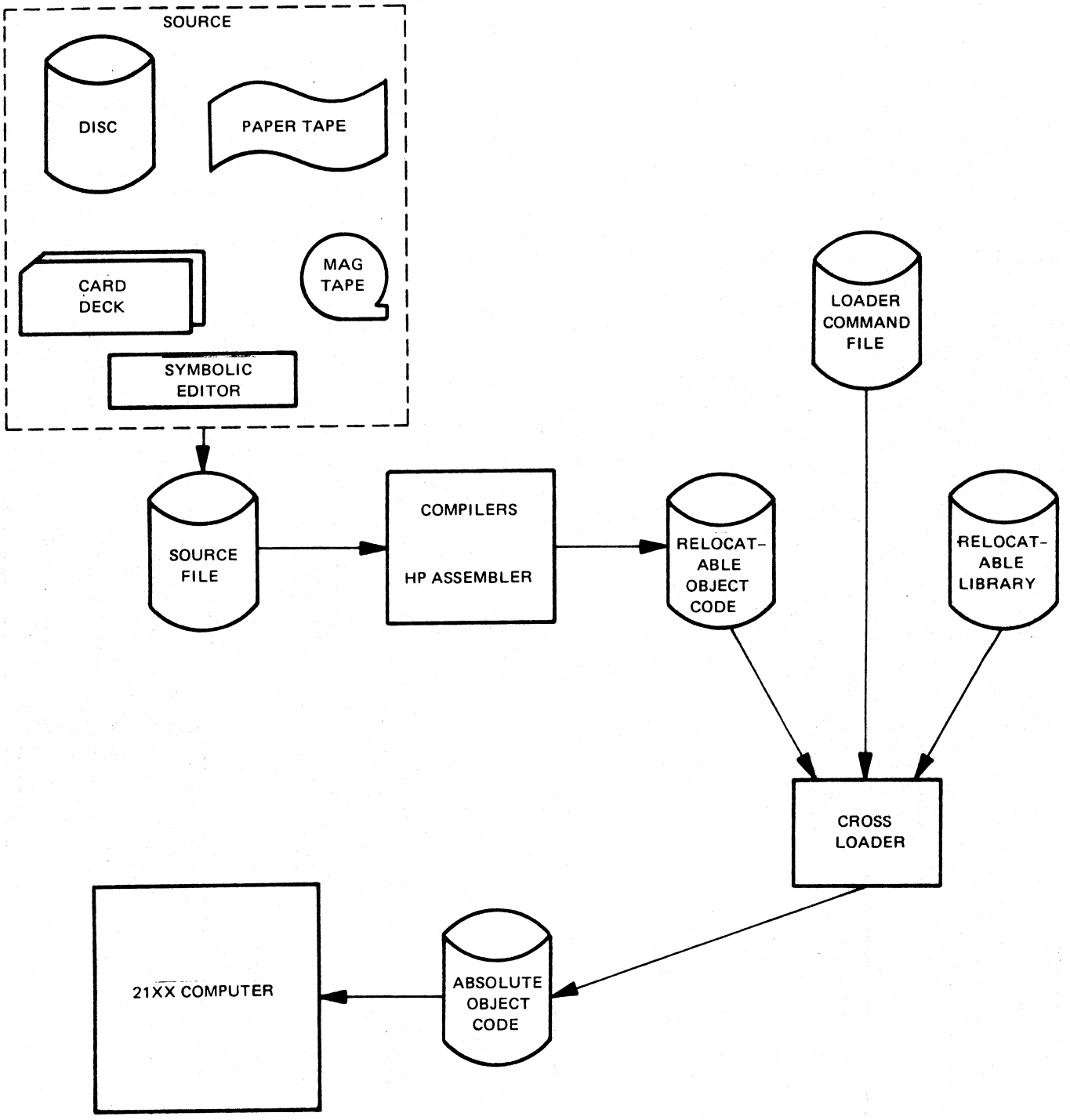


Figure 3-7. Central Program Preparation

The RTE System Cross Loader (SXL) accepts relocatable modules from either an input device, or a file created by the Assembler, ALGOL or FORTRAN Compilers, or load-and-go tracks. The loader links together relocatable program modules, library files and core-resident entry points into executable format. The loader can be used to prepare programs for execution at both the central computer and at terminal computers. A complete description of the SXL Loader is contained in the **RTE System Cross Loader Manual.**

The material which follows assumes that the reader is completely familiar with the **RTE System Cross Loader Manual** and that he wishes to use the SXL Loader specifically in the Distributed System environment.

SXL LOADER USE WITH 91701

The RTE System Cross Loader is used by the Distributed System to relocate code destined for execution in a computer other than the central computer in which the Loader itself runs. To accomplish this, terminal computer configuration parameters must be specified during Phase I of Loader operation. If these parameters are not provided, the Loader will use the indicated default parameters, assuming that the central computer is the target computer. Listed below are the configuration parameters and their default values. All of the assumed values may be redefined by SET or BOUNDS statements.

Core Bounds — FWAM, LWAM, FWABP, LWABP, assumed to be the values for the Background Disc Resident (BKDR) area, but may be set differently by either the BOUNDS or SET statement.

- FWAM – First word available Memory; relocation of modules normally begins here.
- FWABP – First Word Available Base Page. Assumed to be the first word available address for BKDR.
- LWAM – Last Word Available Memory, assumed to be the last available address for BKDR.
- LWABP – LW Base Page – assumed to be 1647_8 (octal).

COMMON — The COMMON area pointers, FWAC and LWAC, default to the contents of LOCC when the first module declaring COMMON is relocated. Note that this means programs are each assumed to require a private area for Common. In order for a program to use the RTE Foreground or Background Common areas, the user must specifically set these pointers (SET or BOUNDS) to the proper value. BKDR programs which attempt to store data into foreground COMMON will violate the memory protect fence; they may "read" data from there, however.

Distributed System user should be aware of two additional points when using the SXL Loader:

- The system library for an EAU (Extended Arithmetic Unit), RTE central computer cannot necessarily be used as the system library for the terminal computer's relocatable binary file.
- Entry points must be defined for the remote monitor system if the Terminal Communications Executive is TCE/3.

The example below shows the commands necessary to relocate modules according to the options set (or defaulted) in Phase 2 of the Loader operation. This example relocates a program from the central computer for a remote BCS terminal computer.

It is assumed that the program TEST makes only simple I/O calls to EXEC, and that an adapter routine named EXEC is available in the library file, so that it is meaningful to relocate this program for execution in a BCS system.

Command	Comments
ON,SXL,1	Turn on the Loader
OUTPUT ABS ON TESTS	Programs to be communicated to a remote must be in absolute binary, BBL-compatible format. This File will be known as TESTS.
BOUNDS FWAM = 2000,& LWAM = 37677,LWABP = 17777,& FWABP = 30,FWAC = 37700,LWAC = & 37677	Define core bounds for target machine
SET .EQT. TO 33721	Define entry points in the resident
SET HALT TO 33701	BCS system of the remote terminal (or use the Loader snap produced when the remote monitor system was generated).
SET XEQT TO 34156	
SET .IOC. TO 33942	
MAP MODULES, GLOBALS	Set memory map flags
RELOCATE TESTR, POWRR, FOURR, RFTR	Relocate the code
SEARCH BCSLIB	Satisfy externals by searching the terminal computer library
DISPLAY UNDEFS	Check that all EXTS are satisfied
NO UNDEFS	Loader output if no UNDEFS exist
END	End of relocation

At this point, a file called TESTS exists on the disc. It can be punched if desired, or if the communication package is available, the remote terminal software may request transfer of the program for execution.

SECTION IV
SYSTEM OPERATION

The Distributed System Interface Kit (91701) provides three levels of Terminal Communications Executives (TCE/1, TCE/2, and TCE/3), any of which can be loaded into a given terminal computer. In multi-terminal systems, the EXEC chosen is matched to a given terminal's task and functions independently of all other terminals.

The terminal executives range in complexity from the TCE/1, 64-word communications loading through the TCE/2 loader with keyboard capability, to the TCE/3 interactive system monitor with interactive system capability.

This Section is divided into parts which describe the system operation of each terminal Executive, plus a summary of the system operation at central.

PART I – TCE/1 SYSTEM OPERATION

This part contains a description of the functions and core requirements of TCE/1. It defines the system operating conventions assumed by this terminal executive, describes its error codes and initialization procedures.

PART II – TCE/2 SYSTEM OPERATION

The system operation of TCE/2 includes a description of the interface between TCE/1 and TCE/2, and system operating conventions. This part also includes the TCE/2 Operator Requests, Error Messages and Initialization procedures.

PART III – TCE/3 SYSTEM OPERATION

This part describes the TCE/3 interfaces and system operating conventions. It includes information on the four kinds of program calls: Remote File Access, Remote EXEC Calls, Local Terminal Utility Calls, and Terminal Access Monitor Calls. The TCE/3 Operator Requests and error codes are provided, together with the initialization procedure for TCE/3.

PART IV – CENTRAL SYSTEM OPERATION

This part contains summaries of the FMP and RTE functions available at the central computer. It includes descriptions of the program calls, operator requests and error messages. The central initialization procedure assumes that the RTE generation (RTGEN) process described in Section VI is complete and that it is the 91701 software which is being made operational.

PART I

TCE/1 SYSTEM OPERATION

INTRODUCTION

The TCE/1 executive is a stand-alone, self-contained loader used in place of the Basic Binary Disc Loader (BBDL), or the Basic Binary Loader (BBL). TCE/1 requires no other subroutines. It resides in the protected, highest 64 words of core. TCE/1 is normally resident in a terminal computer of a Distributed System and is used to load programs selected by 6-bit octal number input through the switch register of the terminal. This executive allows maximum application use of available core. It supports 4K terminal systems.

TCE/1 assumes the following system operating conventions:

- Application program code must be in executable format (absolute) and must reside on the central disc under the name PROGXX. The central computer software prefixes PROG to the switch register input at the terminal and loads PROGXX.
- All I/O drivers for remote communication and instrument peripheral I/O must be provided by the user.
- The communications driver at the terminal (D.65) must be compatible with the communications driver at central (DVR65). See Appendix A for further information.
- All error checks for transfers between the central computer and the terminal must be user written.
- If TCE/1 is required to operate in a BCS environment, it is the user's responsibility to generate the BCS.

There is no provision for servicing program calls or operator requests under TCE/1.

ERROR MESSAGES

TCE/1 returns the single error message shown below for the conditions indicated.

HALT 11B This message indicates that a communications hardware malfunction has occurred. The B-Register returns the following information:

- Ø Parity error or address violation.
- -1 Program file does not exist at central.
- -2 Program is not an absolute binary file, or a read error occurred.

The system aborts.

TCE/1 INITIALIZATION

TCE/1 must be toggled in through the switch register using Appendix E or read in from the paper tape provided.

Application programs and SIO drivers are loaded into terminal core from the central computer following the procedure below:

1. Press HALT.
2. Press SWITCH REGISTER.
3. Set DISPLAY REGISTER to desired program number ($Ø - 77_8$).
4. Press P-REGISTER.
5. Set DISPLAY REGISTER to $n77ØØ_8$.

   where n = Ø - 4K, 1 - 8K, 2 - 12K, 3 - 16K, 5 - 24K, 7 - 32K
6. Press LOADER ENABLE, EXTERNAL PRESET and INTERNAL PRESET.
7. Press RUN.

Computer halts with the following in the DISPLAY REGISTER:

102077_8 = Program loaded

102011_8 = Program not loaded (error)

If 102011_8 HALT, confirm that the program file is on the disc and closed. See TCE/2 error messages in Section IV for additional information.

If 102077 HALT, press P-REGISTER and input the starting address of the program to be run (normally 2_8).

PART II

TCE/2 SYSTEM OPERATION

INTRODUCTION

The TCE/2 is a stand-alone executive which allows application use of all but the uppermost 1K of core, where it resides. It is completely upward compatible with the TCE/3 executive. TCE/2 combines the TCE/1 Loader capability with a functional command repertoire. TCE/2 accepts the operator requests shown from a teletype or equivalent input device. Error messages for this executive are shown in Table 4-1. TCE/2 allows the operator to load and run programs, but does not allow program or file creation or deletion. All application programs to be run at the terminal under TCE/2 may be prepared at the central computer as discussed in Section III of this manual.

TCE/2 assumes the following system operating conventions:

- The communications driver (D.65) supplied by the user at the terminal computer must be compatible with the DVR65 driver at the central computer. (See TCE/2 Generation in Section VI and Appendix A.)

- The TCE/2 executive does not include Basic Control System (BCS), I/O. If it is desirable to use the BCS I/O system at the terminal computer, it is the user's responsibility to generate his own environment. See Section III for more information on Application Program Preparation.

TCE/2 OPERATOR REQUESTS

The TCE/2 operator requests provide direct communication between the operator at the terminal and the 91701 software at the central computer. The table below provides a summary of the requests available in TCE/2.

Table 4-1. Summary of TCE/2 Operator Requests

TCE/2	Function
LOAD	Program load
RUN	Run program at given address
RUN *name*	Load and run program at terminal
RUNAT	Run program at specified address

TCE/2 is a non-interrupt executive which does not automatically recognize keyboard input while a user program is running. Request recognition and service take place automatically at:

TCE/2 initiation
TCE/2 restart
program jump to an EXEC entry point

At these times a colon is displayed and the TCE/2 executive awaits keyboard input. The keyboard syntax for TCE/2 is described below.

Manual interrupt of TCE/2 is accomplished at any time by the following procedures:

1. Press HALT button.
2. Press P-REGISTER.
3. Clear DISPLAY REGISTER.
4. Set DISPLAY REGISTER to 2_8 or $n6400_8$.

   where n = $\emptyset$ - 4K, 1 - 8K, 2 - 12K, 3 - 16K, 5 - 24K, 7 - 32K

5. Press EXTERNAL PRESET, INTERNAL PRESET, and RUN.
6. Input operator request at the display terminal keyboard.

Keyboard input to the TCE/2 is accomplished after the executive returns a colon. TCE/2 waits for an operator request followed by a carriage return. Null input and illegal commands cause SYNTAX ERROR to be displayed, followed by another prompt character.

INPUT REQUEST FORMATS

A blank or a comma delimits a command mnemonic. A comma delimits parameters. Imbedded blanks after the command mnemonic delimiters are ignored. Octal parameters must be preceded with an "@" sign. If an error is made in entering the parameters, a backspace key, or © H, is struck to delete the last character entered. Pressing RUBOUT deletes the entire line. Each request must be terminated with a carriage return.

LOAD

PURPOSE

Load named program at a terminal computer.

FORMAT

LOAD *name*

WHERE

name is the six ASCII character name of the program to be loaded.

COMMENTS

This request reads absolute binary format records from a remote disc file and stores them into terminal core, according to the load address contained in the record. When the request is complete, the terminal executive displays a prompt character (:) and waits for the next command to be entered.

The request is ignored if it contains a syntax error.

SEQUENCE OF OPERATIONS

1. SYNTAX ERROR is displayed if no program name is contained in the request.
2. RLOAD is called to load the program.
3. A WAIT is entered until completion.
4. A check is made for error codes sent from PROGL at central, or on behalf of PROGL. If PROGL was not dormant, STANDBY is output and another call is made to RLOAD.
5. If any errors occur, the appropriate error message is displayed.
6. IDLE is called.

RUN

PURPOSE

Causes loaded program to be executed at a terminal computer, using the starting address loaded into location 3.

FORMAT

RUN [*, *p1*, *p2*, . . . *p5*]

WHERE

p1 . . . *p5* Optional parameter string whose address is passed in the B-Register to the terminal program when the program begins execution.

RUN* , 5, 9, 4ØØ, , 6

or

RUN

COMMENTS

The program to be run is determined by the address last loaded into location 3 when this request is issued.

If this request is issued while a program is running at the terminal, that program is restarted at its initial starting address (assuming no new address has been loaded into location 3).

SEQUENCE OF OPERATIONS

1. Optional parameters are converted to binary.
2. The B-Register is set to the address of the first binary parameter.
3. RNPGM is called to begin execution at the default starting address (location 3).

RUN name

PURPOSE

Causes the named program to be loaded and then executed at a terminal computer.

FORMAT

RUN *name* [,*p1, p2, p3, p4, p5*]

WHERE

name is the ASCII name of the program to be run. This parameter may contain from one to six characters.

p1 . . . p5 an optional parameter string which may be passed to the terminal application program. Each parameter must be a positive integer $\leqslant 32767_{10}$.

RUN MYPROG , , , , , 5

COMMENTS

The address of the binary parameter string is passed to the terminal program in the B-Register when the named program begins execution. If no optional parameters are specified, the B-Register points to a 5-word array of zeros.

SEQUENCE OF OPERATIONS

1. RLOAD is called to load the program and display the appropriate messages if errors occur. If PROGL is not dormant, STANDBY is displayed, and the RLOAD call is repeated until PROGL is dormant.

2. Optional parameters are converted to binary. The B-Register is set to the address of the first binary parameter.

3. RNPGM is called to begin execution at the default start address contained in location 3.

RUNAT

PURPOSE

Causes a program loaded at a terminal computer to be executed at a specified starting address.

FORMAT

RUNAT *nnn* [,*p1*,*p2*, . . . *p5*]

WHERE

nnn	starting address. If the starting address is in octal, it must be preceded with "@."
p1 . . . *p5*	Each parameter is a positive integer value less than or equal to 32767_{10}.

RUNAT @2ØØØ,1,2,3,4,5

COMMENTS

This request assumes that the program to be run has already been loaded as opposed to the RUN *name* request.

SEQUENCE OF OPERATIONS

1. Optional parameters are converted to binary.
2. The B-Register is set to the address of the first binary parameter.
3. RNPGM is called to begin execution at the given starting address.

ERROR MESSAGES

TCE/2 returns the error messages shown in Table 4-2 for the conditions indicated.

Table 4-2. TCE/2 Error Messages

Error Output	Meaning	Action
COM LINE ERR	Communication hardware malfunction. The system aborts.	Reload the system
SYNTAX ERROR	Illegal request	Verify syntax and retry
CAN'T LOCATE	Program to be scheduled at central or loaded at a terminal cannot be found on the central station disc.	Verify program available
PROGL ERROR	File read/write, or close error at a remote computer for a load, or run command. Checksum error.	Verify program type and retry
NO START ADR		Retry with RUNAT request
STANDBY	The central computer is busy at the moment, but will service this request as soon as free.	No additional input is necessary
HALT 13	TCE/2 start address for reconfiguration of I/O channel assignments used without the S-Register being set to new values.	Reset S-Register to correct value. Press RUN.

TCE/2 INITIALIZATION

Load TCE/2 via TCE/1

1. Confirm TCE/1 terminal in protected memory.
2. Press HALT.
3. Press SWITCH REGISTER.
4. Place TCE/2 program number at central in DISPLAY REGISTER.
5. Press P-REGISTER.
6. Place $n7700_8$ in DISPLAY REGISTER

   where n = Ø - 4K, 1 - 8K, 2 - 12K, 3 - 16K, 5 - 24K, 7 - 32K

7. Press LOADER ENABLE, EXTERNAL PRESET, INTERNAL PRESET, and RUN.
8. The computer halts with 102077_8 in the DISPLAY REGISTER when TCE/2 is loaded correctly.
9. If the computer halts with 102011_8 in the DISPLAY REGISTER, TCE/2 is not loaded correctly. Confirm that the program file is on the disc, and that the disc is closed. See error message on the preceding page.

Load TCE/2 via Paper Tape

1. Confirm BBL in protected memory.
2. Press HALT.
3. Press SWITCH REGISTER.
4. Clear DISPLAY REGISTER and place TCE/2 Binary Tape in Paper Tape Reader.
5. Press P-REGISTER.
6. Place $n7700_8$ in DISPLAY REGISTER

   where n = Ø - 4K, 1 - 8K, 2 - 12K, 3 - 16K, 5 - 24K, 7 - 32K

7. Press LOADER ENABLE, EXTERNAL PRESET, INTERNAL PRESET, and RUN.
8. The computer halts with 102077_8 or 102011_8 in the DISPLAY REGISTER when the TCE/2 tape is completed.
9. If the computer halts with 102055_8 or 102011_8 in

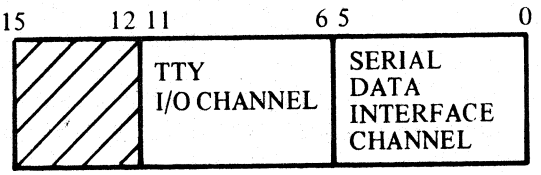
the DISPLAY REGISTER, a Paper Tape Read error has occurred. Retry following Steps 2 - 7 above.

Run TCE/2

1. Press P-REGISTER.
2. Set the DISPLAY REGISTER to the Starting Address of 2_8 (0/000/000/000/000/010)(normal entry).
3. Press INTERNAL PRESET.
4. Press EXTERNAL PRESET.
5. Press RUN.
6. Type in LOAD *name* where *name* is the file name of the application program to be run.

Reconfigure TCE/2 I/O Assignments

1. Press SWITCH REGISTER.
2. Set DISPLAY REGISTER as shown.



3. Press P-REGISTER.
4. Set DISPLAY REGISTER to 100_8.
5. Press EXTERNAL PRESET, INTERNAL PRESET, and RUN.

PART III

TCE/3 SYSTEM OPERATION

INTRODUCTION

TCE/3 is a stand-alone, self-contained application program which resides in low core (see Terminal Core Map in Appendix B). This terminal executive provides four types of program calls to the user: Remote File Access Calls, Remote EXEC Calls, Local Terminal Utility Calls and Terminal Access Monitor Calls. The functions for each call are described in Table 4-3. The Operator Requests available to the TCE/3 user are listed in Table 4-7. The Remote File Access Call Error Messages, the Remote EXEC Call Error Messages and the TCE/3 Request Error Messages returned by TCE/3 are described beginning on page 4-108. The initialization procedures for TCE/3 begin on page 4-111.

TCE/3 assumes the following system operating conventions:

- Application program code may be prepared for the terminal using the System Cross Loader described in Section III.
- Error check after remote calls are user written. Figure 4-1 provides a suggested format for these checks. Coding sequences are discussed in detail in Section III.
- TCE/3 contains a BCS executive so that BCS drivers are compatible.
- The I/O driver for remote communication is provided within the TCE/3 executive.

TCE/3 contains protection features to avoid having itself destroyed by overlay, or from operator interrupt during remote I/O. TCE/3 protects itself from overlay by comparing the load addresses of user programs against TCE/3's .MEM. table. The Last Word of Available Memory of user core is set dynamically during initial startup to the actual core size of the terminal computer. This feature is provided particularly for cases where there are several terminal computers which contain the TCE/3 version of the Terminal Communications Executive, but where the terminal core sizes vary. TCE/3 is generated for the smallest terminal memory, but still allows user programs to make use of the extra core if TCE/3 is loaded into a larger core sized terminal.

TCE/3 protects itself from destruction by operator interrupt by locking out keyboard interrupt processing while remote I/O is in progress.

A complete Remote File Access or Remote EXEC Call communication process is flagged as being in progress by RFAIN via the flag, $BUSY. This flag is located internally, but is declared as an entry point. The flag is used rather than a status call to .IOC. because the driver may go nonbusy periodically during a remote I/O process.

When the TTY/CRT driver (D.ØØ) gets an operator atten-

tion interrupt, it checks $BUSY. If $BUSY is not set, control goes ABORT (the interrupt has been honored). If $BUSY is set, D.ØØ sets the $ESC flag and lets the current operation continue. This means that operator attention has been disabled and the operator must try pressing the key again.

When TEXEC does a program schedule for DLIST and PROGL, it sets $BUSY. $BUSY is cleared when the operation is complete. At key points (between records), TEXEC checks $ESC to see if the operator wishes to abort program-to-program communication. If $ESC is set, a STOP command is transmitted to the remote program (which terminates itself), both flags are cleared and control goes to ABORT.

A terminal application program which initiates direct program-to-program communication must also set and clear $BUSY. If operator attention interrupts are to be allowed during the execution of an application program, the user must check $ESC, and if set; send a STOP, clear flags, and transfer to IDLE. These requirements are due to the fact that TCE/3 has no way of knowing that the application program is doing program-to-program communication.

SUGGESTED INTERFACE CODE FORMAT

The flow chart in Figure 4-1 shows the type of error checks which should be performed for transfers between the terminal and the central computer.

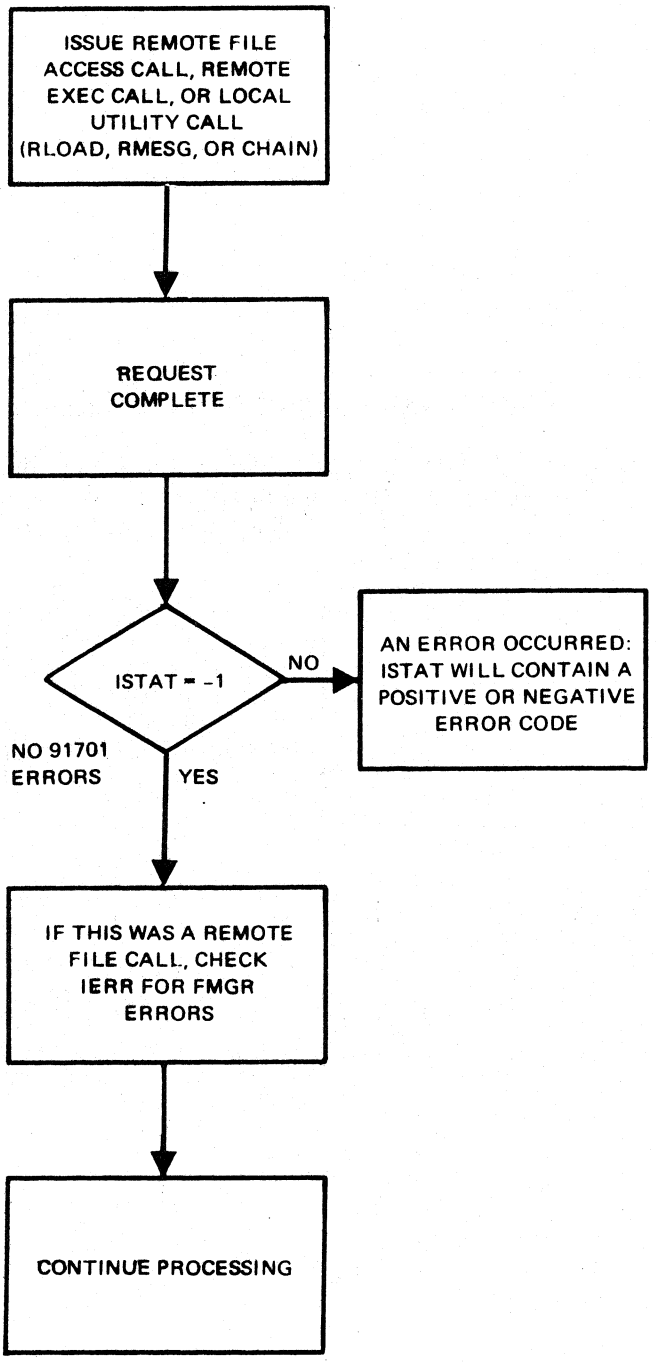


Figure 4-1. Remote Request Error Check

TCE/3 PROGRAM CALLS

TCE/3 provides the user with four kinds of program calls. The Remote File Access program calls interface the terminal user with the Real-Time Executive File Management Package residing at the central computer. The Remote EXEC calls interface the terminal user with the Real-Time Executive System at the central computer. Terminal Access Monitor calls provide the interface between the terminal user and the communication driver. Local Terminal Utility calls are handled by internal modules of TCE/3. Table 4-3 below gives a summary of all of the calls available through TCE/3.

Table 4-3. Summary of TCE/3 Program Calls

Call	Function	Type	Page No.
CHAIN	Load a program from central and execute at a terminal.	Local Utility	4-24
ESCON	Test for ESCAPE during lockout.	Local Utility	4-27
EXECUTION TIME	Schedule a program for execution at central.	REXEC	4-29
EXTENDED PUT, CONVERSATIONAL	Transmit request and receive or transmit data.	#TAM	4-33
GET DATA STREAM	Receive request or data.	#TAM	4-34
GETLU	Obtain Logical Unit at central for this terminal.	Local Utility	4-35
I/O CONTROL	Perform remote I/O operations at central from terminal.	REXEC	4-37
I/O STATUS	Request information on central devices.	REXEC	4-39
IDLE	Return to terminal monitor.	Local Utility	4-41
PROGRAM SCHEDULE	Schedule program at central.	REXEC	4-42
PTPOF	Enable operator attention processing.	Local Utility	4-44
PTPON	Lock out operator attention processing.	Local Utility	4-46
PUT DATA STREAM	Send data buffer to central.	#TAM	4-47
RAPOS	Position a file at central.	RFA	4-48
RCLOS	Close a file at central.	RFA	4-50
RCONT	Perform control function on type Ø file at central.	RFA	4-52
RCRET	Create a file at central.	RFA	4-54
REMOTE READ	Read remote central I/O device from terminal.	REXEC	4-56
REMOTE WRITE	Write remote central I/O device from terminal.	REXEC	4-59
RLOAD	Load program at terminal from central.	Local Utility	4-61
RLOCF	Location and status from DCB.	RFA	4-63

(Continued)

Table 4-3. Summary of TCE/3 Program Calls (Continued)

Call	Function	Type	Page No.
RMESG	Send message from terminal to central display device.	Local Utility	4-65
RNAME	Close DCB and Rename a file at central.	RFA	4-67
RNPGN	Execute loaded program at terminal.	Local Utility	4-69
ROPEN	Open file at central.	RFA	4-71
RPOSN	Position to a record at central.	RFA	4-74
RPURG	Delete a file at central.	RFA	4-76
RREAD	Read from central file.	RFA	4-78
RSTAT	Return DISC Directory.	RFA	4-81
RWIND	Rewind central file.	RFA	4-83
RWRIT	Write a record on central file.	RFA	4-85
STOP	Halt data transfer from central.	#TAM	4-87
TIME REQUEST	Request time from central Real-Time clock.	REXEC	4-88

REMOTE FILE ACCESS CALLS (RFA)

All of the features of the File Management Package are available to the terminal programmer using TCE/3. Files on the central disc may be created, purged, opened and closed. Data may be read or written from or onto these disc files from the terminal program. File positioning calls allow the terminal program to rewind a central disc file or position to a specific record. Terminal programs may obtain status on central files and rename them. Access is also provided to the central computer peripherals such as the line printer, paper tape reader and punch.

File security provided to the terminal programmer is the same as that provided to the File Management Package user at central:

- Each file has a security code.
- Security codes may be positive, negative or zero.
- If the file code is positive and the code at open does not match, the file may be read but not written on.
- If the file code is negative and the code used at open does not match, the file will not be opened.
- If the file code is zero, the file may be opened to any caller with no restrictions.

File types available to the terminal user are the same as those provided by FMP to the central programmer, except zero and negative record lengths, which are not supported at the terminal. Table 4-4 below provides a summary of these types and a brief description of their usage. For more information, the **Real-Time Executive File Manager Manual** should be consulted.

Table 4-4. Remote File Access File Types

Type	Description
0	This file type can only be created at the central computer. The file directory entry is located in the system disc file directory and has special entries relating to the logical unit number and end-of-file code.
1	Type 1 files have fixed record lengths of 128 words. Transfers to or from type 1 files are done directly to or from the user's buffers. Any file type except type Ø may be opened as a type 1 file. It is a fixed length, random access, and non-extendable file.

(Continued)

Table 4-4. Remote File Access File Types (Continued)

Type	Description
2	Type 2 files have user defined record lengths to a maximum of 128 words. Each transfer is one record long, and must go through a packing buffer. Type 2 files are fixed length, random access and non-extendable.
3	Type 3 files are packed on the central disc and contain data, source, relocatable or absolute information. Each transfer is one record long and must go through a packing buffer. These files are random length to a maximum of 128 words, sequential access with automatic extents.
4	These files have the same characteristics as type 3 files, except FMP recognizes type 4 files as containing source programs. Type 4 files are random length to a maximum of 128 words, sequential access, with automatic extents.
5	These files have the same characteristics as type 3 files, except that FMP recognizes type 5 files as containing relocatable binary code. These files are random length, to a maximum of 128 words, sequential access, with automatic extents.
6	These files have the same characteristics as type 3 files. These files are created as type 6, but always accessed as type 1. Type 6 files are random length, to a maximum of 128 words, sequential access, with automatic extents.
7	Type 7 files have the same characteristics as type 3 files, except that FMP recognizes these files as containing absolute binary code. Type 7 files are random length, sequential access, with automatic extents.
>7	Any file type greater than 7 is a user defined file with the same characteristics as type 3 files. FMP will treat these files as random length files with sequential access and automatic extents.

Remote File Access Calls to the File Management Package at the central computer employ a common set of parameters as part of their calling sequence. These common parameters are described below. Parameters that are unique to a call are described within the call itself. It should be noted, if not already obvious from the names, that File Management Calls issued from a terminal are in a different format from File Management calls issued at the central computer. It should also be noted that the TCE/3 user is not responsible for supplying or pointing to the 144 - DCB as is the case with the FMP user at the central computer.

ISTAT — Contains the communication status returned by TCE/3 remote central computer. Positive codes are returned **before** transmission begins, negative codes **after** transmission begins.

Legal values for this parameter are:

5	RFA busy, or maximum number of remote open files exceeded
4	Not used
3	Not enough parameters
2	Communications line down
1	Illegal request code
Ø	Request being processed
-1	Request completed, no errors
-2	Not used
-3	I/O error
-4	Error in parameters, or program to be scheduled does not exist
-5	Improper sequence of commands used (file not open to terminal)

IERR — The status of the remote file operation returned from the file manager.

Possible values are:

≥Ø	None
-1	Disc down
-2	Duplicate name
-3	Backspace Not Legal (Type Ø)
-4	File too long or REC Size Error
-5	Attempt to read or position to a record not written, or on update, write an illegal length
-6	Cartridge not found or file not found or no room
-7	Invalid security code
-8	File currently open: eight PGM or exclusive or LOCK rejected
-9	Attempt to open type Ø as type 1 or to use APOSN on type Ø

-10 Not enough parameters
-11 DCB not open
-12 SOF or EOF read or sensed
-13 Cartridge locked
-14 Directory full
-15 Illegal name
-16 Illegal Type or Size = Ø (Creat)
-17 Attempt to Read or Write on type Ø which does not support the operation

NAME — ASCII name of file to be accessed. This parameter can be up to six characters in length. If less than six characters, trailing blanks are required.

READ/WRITE BUFFER SIZE — The maximum record size is 128 words for remote file access calls. Zero and negative signs are illegal.

Sequence of Operations for TCE/3 Remote File Access Calls

1. Fetch caller's parameters.
2. Check for enough parameters.
3. Check for legality of parameters.
4. Generate a PARMB request buffer which includes identification and function codes plus all of the user parameters necessary to reconstruct this call at the central computer.
5. Send PARMB to central by issuing a PUT DATA STREAM call to # TAM. This results in a TRANSMIT REQUEST ONLY call to .IOC. If the call is to read or write a data record, a PUT CONVERSATIONAL call is issued to # TAM instead of the PUT DATA STREAM call. The PUT CONVERSATIONAL call results in a TRANSMIT REQUEST AND RECEIVE OR TRANSMIT DATA call to .IOC.
6. Wait for completion, and check for D.65/DVR65 I/O errors.
7. The calling sequence is reconstructed at central and a local call is issued to FMP. A DCB is attached by the 91701 software at central before the local call is issued.
8. See the **Real-Time Executive File Manager System Manual** for the sequence of operations during the FMP call execution.
9. Receive the reply buffer by issuing a GET DATA STREAM call to # TAM. This results in a RECEIVE REQUEST ONLY call to .IOC. Polling mode is pro-

vided by # TAM, which re-issues the .IOC. Call until the reply is received.

10. A check is made for a legal reply.

11. Transfer return parameters from the reply buffer to the user call parameters. Fetch the A-Register and B-Register values from the reply buffer.

12. Return to the caller with the A-Register and B-Register at the terminal mirroring the A- and B-Register values at central when the call was executed.

REMOTE EXEC CALLS (REXEC)

A subset of the standard RTE EXEC calls are available to the terminal programmer to perform functions at the central computer. These calls, in modified formats, call the TCE/3 programmer at the terminal to schedule program execution at the central computer. The terminal program may read or write data on remote I/O devices at the central computer. Remote I/O control calls allow the terminal program to rewind and position magnetic tape, punch paper tape leaders, and control line printer spacing. Terminal programs may obtain identity and status of central computer peripherals using these calls.

Sequence of Operations for TCE/3 Remote EXEC Calls

1. Fetch the caller's parameters.

2. Check for enough parameters.

3. Check parameter legality.

4. Generate a PARMB request buffer, which includes identification and function codes, plus all user parameters necessary to reconstruct this call at the central computer.

5. Send PARMB to central by issuing a PUT DATA STREAM call to # TAM. This results in a TRANSMIT REQUEST ONLY call to .IOC. If the call is to read or write a data record, a PUT CONVERSATIONAL call is issued to # TAM instead of the PUT DATA STREAM. The PUT CONVERSATIONAL results in a TRANSMIT REQUEST AND RECEIVE OR TRANSMIT DATA call to .IOC.

6. Wait for completion, and check for D.65/DVR65 I/O errors.

7. The calling sequence is reconstructed at central and a local call is issued to RTE.

8. See the **Real-Time Executive Software System Manual** for the sequence of operations during the RTE call execution.

9. Receive reply buffer by issuing a GET DATA STREAM call to #TAM. This results in a RECEIVE REQUEST ONLY call to .IOC. Polling mode is provided by #TAM, which re-issues the .IOC. call until the reply is received.

10. A check is made for a legal reply.

11. Transfer return parameters from the reply buffer to the user call parameters. Fetch the A-Register and B-Register values from the reply buffer.

12. Return to the caller with the A- and B-Registers at the terminal set to the same values the A- and B-Registers at central contained when the local call at central was executed.

LOCAL TERMINAL UTILITY CALLS

A set of utility calls are provided to the TCE/3 terminal programmer to facilitate local terminal functions. These calls provide the ability to load a program from the central computer and execute it at the terminal, to test for operator interrupt during remote I/O processing, to obtain the Logical Unit Number assigned to this terminal at the central computer and to send messages to the central computer display device.

The local utility functions are implemented by making subroutine calls to entry points in the Terminal Executive (TEXEC) Module of TCE/3. Some of these calls result in Remote File Access calls or Remote EXEC calls to RFAIN, and some of the calls are processed directly by TEXEC. The table below shows a summary of the local utility calls and the resulting remote calls, if applicable.

Table 4-5. Summary of Local Terminal Utility Remote Calls

TEXEC Utility	Resulting Remote Call
CHAIN	REXEC (schedule PROGL)
ESCON	None
GETLU	#TAM
IDLE	None
PTPON	None
PTPOF	None
RLOAD	REXEC (schedule PROGL)
RMESG	REXEC (write)
RNPGM	None

TERMINAL ACCESS MONITOR CALLS (# TAM)

A set of program calls are provided to the 91701 Terminal Access Monitor (TAM) for remote communication. TAM performs this communication via .IOC. to D.65. Polling mode is provided to receive "request" calls. The transmission mode is determined internally with TAM. For non-DMA terminals, the transmission mode is "interrupt open loop." Transmission mode for DMA terminals is "DMA open loop." I/O is performed in privileged mode. This module is available to the terminal user for the functions shown in Table 4-6. # TAM calls are not FORTRAN callable.

Status codes are returned according to the success of the operation. On entry to the completion subroutine, or upon return to the caller, if the operation is completed immediately, the A-Register contains the following:

- -3 Communication I/O error
- -1 Request completed, no error
- Ø Request being processed
- 1 Illegal TAM request code
- 2 Communications line down or transmission error
- 3 Not enough parameters in TAM call

Table 4-6. Summary of Terminal Access Monitor Calls

# TAM Function Codes	# TAM Function	D.65 Function
2	STOP	Transmit STOP Reply
5	Get Data Stream	Receive Request Only or Receive Data only
6	Put Data Stream	Transmit Request Only or Transmit Data Only
8	Extended Put, conversational	Transmit Request and Receive or Transmit Data

Appendix A is provided for the user who wishes to provide his own communication link via .IOC. to D.65. Table 4-6 shows the correspondence between # TAM calls and .IOC. calls to D.65.

CHAIN

PURPOSE

Load a program from the central computer into a terminal and execute at the terminal (terminal program may overlay itself).

```
        EXT     CHAIN
         .
         .
         .
        JSB     CHAIN     Transfer control to TCE/3
        DEF     RTN       Point of return if an error occurs
        DEF     ISTAT     Communication status
        DEF     IERR      Operational status
        DEF     IPROG     Program to load and execute
        DEF     IADR      Optional start address (see Comments)
        DEF     p1        Optional user parameters passed to loaded program
         .
         .
         .
        DEF     p5
RTN     return point      If control reaches this point, an error has occurred.
         .
         .
         .
ISTAT   BSS     1         Status code returned here (see Comments).
IERR    BSS     1         Error code returned here (see Comments).
IPROG   ASC     3,name    Program name.
IADR    OCT     address   Starting address. If a value is not supplied for IADR, the
                          loaded program begins execution at the contents of absolute
                          location 3.
p1      BSS     1         The B-Register points to this parameter string when control is
 .       .      .         passed to the loaded program.
 .       .      .
 .       .      .
p5      BSS     1
```

COMMENTS

If the starting address is not known and optional parameters are desired, IADR is set to 2.

Legal values for the codes returned in status for this call are:

- 5 = RFA busy
- 4 = Not used
- 3 = Not enough parameters
- 2 = Communications line down
- 1 = Illegal request code
- ∅ = Request being processed
- -1 = Request completed, no errors
- -2 = Not used
- -3 = I/O error
- -4 = Error in parameters
- -5 = Improper sequence of commands

Legal error codes returned in IERR for this call are:

- ∅ = No error
- -1 = Open error on program name (the file probably does not exist on the central disc).
- -2 = File read error at remote
- -3 = Transmission error in data record
- -4 = Transmission error between central and terminal computers

-5 = File close error at remote

-6 = Start address not available (no default address provided in location 3)

If no optional parameters are passed in this call, the B-Register points to a 5-word array of zeros when control is passed to the loaded program.

SEQUENCE OF OPERATIONS

1. Get the parameters from the user calling sequence.
2. Call RLOAD to load the program.
3. Call RNPGM to execute the program.
4. If no starting address is specified in the CHAIN call and no defaults exist, return to the caller.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAM AND ALGOL WILL CAUSE THE SEGMENT 'SEG3' TO BE OVERLAYED BY THE SEGMENT 'SEG4' AND WHILE SPECIFYING THAT 'SEG4'S ABSOLUTE STARTING ADDRESS WILL BE FOUND IN ABSOLUTE MEMORY LOCATION 3 AFTER 'SEG4' IS LOADED,THE FILE NAME 'WORKER' WITH SECURITY CODE 'JB' ON CARTRIDGE '9701' AT CENTRAL WILL BE PASSED TO 'SEG4' AS PARAMETERS AFTER IT OVERLAYS 'SEG3' AT THE TERMINAL .

----------FORTRAN----------

```
FTN,L
      PROGRAM SEG3
      DIMENSION SEG4(3)
      DATA SEG4/2HSE,2HG4,2H  /
      DATA IPRM1/2HWO/,IPRM2/2HRK/,IPRM3/2HER/,IPRM4/2HJB/,IPRM5/9701/
       .
      CALL CHAIN(ISTAT,IERR,SEG4,0,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5)
      IF (ISTAT.........
      IF (IERR.....
      END
       .
      END$

FTN,L
      PROGRAM SEG4
      DIMENSION IPRAM(5)
       .
      CALL RMPAR(IPRAM)
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"SEG3"
    BEGIN
       .
  INTEGER ARRAY SEG4[1:3] := "SE","G4","  " ;
  INTEGER ISTAT,IERR ;
       .
PROCEDURE CHAIN(ISTAT,IERR,IPROG,IADR,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5) ;
    VALUE IADR,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
  INTEGER ISTAT,IERR,IPROG,IADR,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
    CODE ;
       .
```

```
        CHAIN(ISTAT,IERR,SEG4[1],0,"WO","RK","ER","JB",9701) ;
        IF ISTAT.............. ;
        IF IERR.............. ;
    END$

HPAL,L,"SEG4"
  BEGIN
      .
  INTEGER IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
      .
PROCEDURE RMPAR(P) ;
  INTEGER P ;
    CODE ;
      .
        RMPAR(IPRM1) ;
      .
    END$
```

ESCON

PURPOSE

Test for ESCAPE key during lockout.

```
        EXT     ESCON
         .
         .
         .
        JSB     ESCON
        DEF     * + 2
        DEF     IND         ESCAPE key indicator
         .
         .
         .
IND     BSS     1           IND = ∅ if key not pressed.
                            IND = -1 if key pressed.
```

COMMENTS

When the test indicates the key was pressed, the terminal program may want to terminate the central program via the STOP command. This call is used in conjunction with PTPON/PTPOF during user program-to-program communication. See PTPON call and TAM (STOP) call.

SEQUENCE OF OPERATIONS

1. Get $ESC flag.
2. If the flag is zero, the ESCAPE was not pressed, and a zero is returned to the caller.
3. If $ESC is not zero (the ESCAPE key was pressed), a minus one is returned to the caller.
4. Return.

FORTRAN & ALGOL EXAMPLES

```
----------FORTRAN----------

FTN,L
      PROGRAM REMOT
       .
      CALL ESCON(I)
      IF (I.EQ.-1)CALL IDLE
       .
      END
       .
      END$

----------ALGOL----------
HPAL,L,"REMOT"
   BEGIN
     .
  INTEGER I ;
     .
PROCEDURE ESCON(I) ;
  INTEGER I ;
    CODE ;
```

```
PROCEDURE IDLE ;
     CODE ;
          •
          ESCON(I); IF I THEN IDLE ;
          •
       END$
```

EXECUTION TIME (Initial Offset Version)

PURPOSE

To schedule a program for execution at the remote computer at specified time intervals. Execution may be scheduled after an initial offset time (Initial Time Version), or at a particular absolute time (Absolute Start-Time Version). RTE places the requested program in the time list and returns to the terminal calling program.

```
        EXT     REXEC
         .
         .
         .
        JSB     REXEC     Transfer control to remote RTE
        DEF     RTN       Point of return
        DEF     ISTAT     Communication status
        DEF     ICODE     Request code
        DEF     IPROG     Remote program to put into time list
        DEF     IRESL     Resolution code
        DEF     MTPLE     Execution multiple
        DEF     IOFST     Initial time offset
RTN     return point      Continue execution.
         .
         .
         .
ISTAT   BSS     1         Communication status returned here.
ICODE   DEC     12        Request code = 12.
IPROG   ASC     3,xxxxx   xxxxx is the name of the program to put in time list.
IRESL   DEC     x         x is the resolution code (see Comments).
MTPLE   DEC     y         y is the execution multiple (see Comments).
IOFST   DEC     -z        z (units set by x) gives the initial offset; the negative sign signals
                          the initial offset version of this call to RTE.
```

COMMENTS

The resolution code (x) represents the units of time, which when used with the execution interval value, results in the total time interval. Legal values for this parameter are:

1 - tens of milliseconds

2 - seconds

3 - minutes

4 - hours

The execution multiple (y) is a number between Ø and 999 which is used with x to give the actual time interval.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILLQUEUE THE CENTRAL PROGRAM 'QUIKR' TO BE SCHEDULE'D TEN(10) MILLISECONDS AFTER THE CALL IS RECEIVED,AND SPECIFY THAT IT BE RE-SCHEDULE'D EVERY FIVE MINUTES THEREAFTER. NOTE - 5 MIN := 100 * 60 * 5 := 30000 MILLISEC

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
      DIMENSION QUIKR(3),IREG(2)
      EQUIVALENCE (IREG(2),REG,IA),(IREG(2),IB)
      DATA QUIKR/2HQU,2HIK,2HR /
       .
      REG = REXEC(ISTAT,12,QUIKR,1,30000,-1)
      IF (ISTAT.............
      IF (IA..........
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REXUS"
    BEGIN
       .
  INTEGER ARRAY QUIKR[1:3] := "QU","IK","R " ;
  INTEGER ISTAT,IA,IB
       .
     REAL
PROCEDURE REXEC(ISTAT,ICODE,IPROG,IRESL,MTPLE,IOFST) ;
    VALUE ICODE,IRESL,MTPLE,IOFST ;
  INTEGER ISTAT,ICODE,IPROG,IRESL,MTPLE,IOFST ;
    CODE ;
PROCEDURE SEPAR(P1,P2,P3) ;
    VALUE P1 ;
     REAL P1 ;
  INTEGER P2,P3 ;
    CODE ;
         .
         SEPAR(REXEC(ISTAT,12,QUIKR[1],1,30000,-1),IA,IB) ;
         IF ISTAT.............. ;
         IF IA................ ;
         .
      END$
```

EXECUTION TIME (Absolute Start-Time Version)

PURPOSE

The absolute Start-Time Version of the Execution Time call is the same as the Initial Offset Version up to the IOFST parameter. The sign of IOFST must be positive. The positive number in the parameter signals RTE to expect three more parameters.

```
        EXT     REXEC
         .
         .
         .
        JSB     REXEC   Transfer control to remote RTE
        DEF     RTN     Point of return
        DEF     ISTAT   Communication status
        DEF     ICODE   Request code
        DEF     IPROG   Remote program to put into time list
        DEF     IRESL   Resolution code
        DEF     MTPLE   Execution multiple
        DEF     IHRS
        DEF     MINS    Absolute start time on a 24-hour clock
        DEF     ISECS
        DEF     MSECS
RTN     return point
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
ICODE   DEC     2       Request code = 12.
IPROG   ASC     3,xxxxx xxxxx is the name of the program put in time list.
IRESL   DEC     x       x is the resolution code.
MTPLE   DEC     y       y is the execution multiple.
IHRS    DEC     a       Absolute starting time, in hours,
MINS    DEC     b         minutes,
ISECS   DEC     c         seconds,
MSECS   DEC     d         and tens of milliseconds on a 24-hour clock.
```

COMMENTS

The resolution code (x) represents the units of time, which when used with the execution interval value, results in the total time interval. Legal values for this parameter are:

1 - tens of milliseconds
2 - seconds
3 - minutes
4 - hours

The execution multiple (y) is a number between Ø and 999 which is used with x to give the actual time interval.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL QUEUE THE CENTRAL PROGRAM 'QUIKR' TO BE SCHEDULE'D AT MIDNITE EACH DAY.

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
      DIMENSION QUIKR(3),IREG(2)
      EQUIVALENCE (IREG(2),REG,IA),(IREG(2),IB)
      DATA QUIKR/2HQU,2HIK,2HR /
       .
      REG = REXEC(ISTAT,12,QUIKR,4,24,0,0,0,0)
      IF (ISTAT.............
      IF (IA..........
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REXUS"
    BEGIN
          .
  INTEGER ARRAY QUIKR[1:3] := "QU","IK","R " ;
  INTEGER ISTAT,IA,IB
          .
     REAL
PROCEDURE REXEC(ISTAT,ICODE,IPROG,IRESL,MTPLE,IHRS,MINS,ISECS,MSECS) :
    VALUE ICODE,IRESL,MTPLE,IHRS,MINS,ISECS,MSECS ;
  INTEGER ISTAT,ICODE,IPROG,IRESL,MTPLE,IHRS,MINS,ISECS,MSECS ;
    CODE ;
PROCEDURE SEPAR(P1,P2,P3) ;
    VALUE P1 ;
     REAL P1 ;
  INTEGER P2,P3 ;
    CODE ;
          .
          SEPAR(REXEC(ISTAT,12,QUIKR[1],4,24,0,0,0,0),IA,IB) ;
          IF ISTAT.............. ;
          IF IA................ ;
          .
      END$
```

EXTENDED PUT, CONVERSATIONAL

PURPOSE (Internal TCE Call)

Transmit request and receive or transmit data. CAUTION: This is a special call used by TCE/3 for remote file read/write requests and is included here for reference only. User programs must not make this call.

```
        EXT     #TAM
         .
         .
         .
(A)  =  Ø for receive data
     =  1 for transmit data

        JSB     #TAM    Transfer control to TAM
        DEF     *+7     Return address
        DEC     8       Function code
        DEF     q       Request buffer address
        DEC     r       Request buffer length (negative bytes)
        DEF     s       Data buffer address
        DEC     t       Data buffer length (negative bytes)
        DEF     u       Completion subroutine
         .
         .
         .
q       BSS     n       Request buffer.
s       BSS     m       Data buffer.
u       NOP             Routine to which a JSB is performed when I/O is complete.
```

COMMENTS

This call is used only within TCE/3. It causes an unsolicited interrupt at central for terminal remote file access requests. The request buffer is of a special format that is known to the central QUEUE module.

SEQUENCE OF OPERATIONS

1. Check the number of parameters.
2. Format .IOC. call using the caller's buffer addresses and buffer lengths.
3. Format for .IOC. call and set up for a Transmit Request and Receive or Transmit data call.
4. Issue .IOC. call.
5. Wait for completion.
6. Set the A-register to minus one, or the appropriate #TAM error code.
7. Return.

FORTRAN & ALGOL

This call cannot be made using FORTRAN or ALGOL.

GET DATA STREAM

PURPOSE

This call is used at the terminal to receive the request or the data transfer initiated by a Transmit Request or Transmit Data call at central. The request/data buffer length at the terminal must be exactly the same as the request/data buffer length at central. See the Terminal Receives Data program example in Section VII.

```
        EXT     #TAM
         .
         .
         .
(A)  =  Ø for receive data
     =  1 for receive request
        JSB     #TAM    Transfer control to TAM
        DEF     * + 5   Return address
        DEC     5       Function code
        DEF     x       Request or data buffer
        DEC     y       Buffer length (negative bytes)
        DEF     z       Completion subroutine
         .
         .
         .
x       BSS     n       Request or data buffer.
z       NOP             Routine to which a JSB is performed when I/O is complete.
         .
         .
         .
```

SEQUENCE OF OPERATIONS

1. Check the number of parameters.
2. Format .IOC. call using the caller's buffer address and buffer length.
3. If "get request," enable the polling mode.
4. Set up mode for .IOC. call, and set up for receive request or receive data.
5. Issue .IOC. call.
6. Wait for completion.
7. Set the A-Register to minus one, or the appropriate #TAM error code.
8. Return.

FORTRAN & ALGOL

This call cannot be made using FORTRAN or ALGOL.

GETLU

PURPOSE

Obtain the logical unit at central for this terminal.

```
        EXT     GETLU
         .
         .
         .
        JSB     GETLU
        DEF     * + 2
        DEF     LU
         .
         .
         .
LU      BSS     1       Logical Unit number (returned).
```

SEQUENCE OF OPERATIONS

1. Set $BUSY flag, if not already set.
2. Send a request to central to return the logical unit number for this terminal.
3. Receive request reply from central.
4. Check for errors; if any, try again.
5. Clear $BUSY flag if it was clear on entry into GETLU.
6. Fetch the returned logical unit number from the reply buffer and pass it to the user.
7. Return.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING FORTRAN AND ALGOL EXAMPLES WILL DETERMINE PROGRAMMATICALLY THE LOGICAL UNIT NUMBER OF THE CALLING TERMINAL AT CENTRAL.

```
--------------------------------FORTRAN---------------------------------

FTN,L
      PROGRAM REMOT
       .
      CALL GETLU(LU)
       .
      END
       .
      END$

--------------------------------ALGOL-----------------------------------

HPAL,L,"REMOT"
    BEGIN
        .
  INTEGER LU ;
        .
PROCEDURE GETLU(LU) ;
  INTEGER LU ;
    CODE ;
        .
```

```
    GETLU(LU) ;
.
END$
```

I/O CONTROL

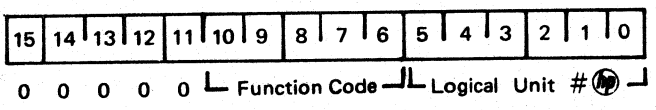
PURPOSE

To carry out various remote I/O control operations at central such as backspace, write end-of-file, rewind, etc.

```
        EXT     REXEC
         .
         .
         .
        JSB     REXEC   Transfer control to remote RTE
        DEF     RTN     Point of return
        DEF     ISTAT   Communication status
        DEF     ICODE   Request code
        DEF     ICNWD   Control information
        DEF     IPRAM   Optional parameter
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
ICODE   DEC     3       Request code = 3.
ICNWD   OCT     conwd   See Control Word.
IPRAM   DEC     n       Required to list output line spacing (see Control Word).
```

CONTROL WORD

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



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

The following functions are defined for DVR00, DVR01, and DVR02:

20	Enable auxiliary TTY-type device at central to schedule its programs when any key is struck.
21	Disable auxiliary TTY-type device at central to inhibit scheduling of programs.
22	Set time out. The optional third parameter is set as the new time-out interval.
23	Ignore all further action requests until one of the following occurs: a. The queue is empty, b. An interrupt request is received, c. A restore control request is received.
24	Restore output processing.

Function Code 11_8 (list output line spacing) requires the optional parameter IPRAM. IPRAM must designate the number of lines to be spaced on the specified logical unit. A negative parameter specifies a page eject on a line printer.

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

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL SKIP LEADER ON THE STANDARD INPUT <ICNWD=5> AT CENTRAL,THEN WILL GENERATE A TOP-OF-FORM REQUEST FOR THE STANDARD LIST DEVICE <ICNWD=6> ,ALSO AT CENTRAL.

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
       .
      CALL REXEC(ISTAT,3,5+700B)
      IF (ISTAT.............
       .
      CALL REXEC(ISTAT,3,6+1100B,-1)
      IF (ISTAT............
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REXUS"
    BEGIN
          .
PROCEDURE REXEC(ISTAT,ICODE,ICNWD,IPRAM) ;
    VALUE ICODE,ICNWD,IPRAM ;
  INTEGER ISTAT,ICODE,ICNWD,IPRAM ;
     CODE ;
          .
          REXEC(ISTAT,3,5+@700,0) ;
          IF ISTAT.....................;
          .
          REXEC(ISTAT,3,6+@1100,-1) ;
          IF ISTAT.....................;
          .
      END$
```

I/O STATUS

PURPOSE

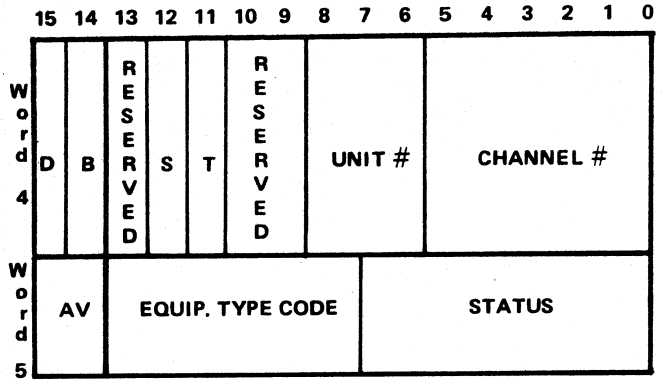
Request information (status condition and device type) about a device assigned to a remote logical unit number.

```
        EXT     REXEC
        .
        .
        .
        JSB     REXEC     Transfer control to remote RTE
        DEF     RTN       Point of return
        DEF     ISTAT     Communication status
        DEF     ICODE     Request code
        DEF     ICNWD     Control information
        DEF     ISTA1     Status word 1
        DEF     ISTA2     Status word 2 (optional)
RTN     return point      Continue execution
        .
        .
        .
ISTAT   BSS     1         Communication status returned here.
ICODE   DEC     13        Request code = 13.
ICNWD   DEC     n         Remote Logical Unit number.
ISTA1   NOP               Word 5 of remote EQT entry returned here.
ISTA2   NOP               Word 4 of remote EQT entry returned here.
```

COMMENTS

The second status word is optional in the I/O status EXEC Call.

Words 4 and 5 of EQT entry table contain the following information:



Where:

- D = 1 if DMA is required.
- B = 1 if automatic output buffering is used.
- S = 1 if a driver is to process time-out on this device.
- T = 1 if the device time out (System sets to zero before each I/O request).
- UNIT = Last sub-channel addressed.
- CHANNEL = I/O select code for device (lower number if a multi-board interface).
- AV = Availability indicator:
  - 0 = Available for use
  - 1 = Disabled (down)
  - 2 = Busy (currently in operation)
  - 3 = Waiting for an available DMA channel

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

- 00 to 07_8 = paper tape devices (or system control devices)
  - 00 = teleprinter (or system keyboard control device)
  - 01 = photo reader
  - 02 = paper tape punch
- 10 to 17 = unit record devices
  - 10 = plotter
  - 12 = line printer
- 20 to 37 = magnetic type/mass storage devices
  - 30 = fixed head disc or drum
  - 31 = moving head disc
- 40 to 77 = instruments

STATUS = the actual physical status or simulated status at the end of each operation.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL OBTAIN THE DEVICE TYPE OF THE STANDARD LIST DEVICE AT CENTRAL.

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
       .
      CALL REXEC(ISTAT,13,6,ISTA1)
      IF (ISTAT..............
      ITYPE = IAND(ISTA1,037400B)  / 400B
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REXUS"
    BEGIN
       .
  INTEGER ISTAT,ISTA1,ITYPE ;
       .
PROCEDURE REXEC(ISTAT,ICODE,ICNWD,ISTA1,ISTA2) ;
    VALUE ICODE,ICNWD ;
  INTEGER ISTAT,ICODE,ICNWD,ISTA1,ISTA2 ;
    CODE ;
       .
        REXEC(ISTAT,13,6,ISTA1,ISTA2) ;
        IF ISTAT.......... ;
        ITYPE := ISTA1 AND @037400 \ @400 ;
       .
      END$
```

IDLE

PURPOSE

To return to the terminal monitor from a user program.

```
EXT     IDLE
 .
 .
 .
JSB     IDLE      Transfer control to TCE/3.
DEF     * + 1     The prompt character will be output at the TTY, and TCE/3
                  will then accept the next operator command.
```

COMMENTS

Every terminal user program should terminate via a call to IDLE to preserve the continuity of the TCE/3 system.

SEQUENCE OF OPERATIONS

1. Display ":" prompt.
2. Continue in loop waiting for next operator request to be input.

FORTRAN & ALGOL EXAMPLES

```
----------------------------------FORTRAN----------------------------------

FTN,L
      PROGRAM REMOT
        .
      CALL IDLE
        .
      END
        .
      END$

----------------------------------ALGOL------------------------------------

HPAL,L,"REMOT"
    BEGIN
        .
PROCEDURE IDLE ;
     CODE ;
         .
          IDLE ;
         .
       END$
```

PROGRAM SCHEDULE

PURPOSE

Schedule a dormant remote program for execution at the central (remote) computer, and optionally to transfer up to five parameters to that program.

```
        EXT     REXEC
         .
         .
         .
        JSB     REXEC
        DEF     RTN       Transfer control to remote RTE
        DEF     ISTAT     Communication status
        DEF     ICODE     Request code
        DEF     INAME     Name of remote program to schedule
        DEF     IPRM1  ]
         .              }  Up to five optional parameters
         .              }
         .              }
        DEF     IPRM5  ]
RTN     return point      Continue execution
         .
         .
         .
ISTAT   BSS     1         Communication status returned here.
ICODE   DEC     10        Request code = 10. No wait.
INAME   ASC     3,xxxxx   xxxxx is the name of the program to schedule.
IPRM1  ]
       }                  User specified 5 entry array, values may be positive or negative.
IPRM5  ]
```

COMMENTS

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

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

1 = Scheduled

2 = I/O suspend

3 = Not used

4 = Unavailable memory suspend

5 = Disc allocation suspend

6 = Operator suspend, or programmed suspend (EXEC7 Call).

PARAMETERS

When the scheduled program begins executing, the B-Register contains the address of a five-word list of parameters from the calling program (the parameters equal zero if none were specified.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY SCHEDULE THE FILE MANAGER AT CENTRAL, AND PASS TO IT THE NAME OF A TRANSFER FILE 'PACK' WHICH CONTAINS FILE MANAGER INSTRUCTIONS TO PACK THE DISC.

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
      DIMENSION INAME(3),IREG(2)
      EQUIVALENCE (IREG(1),REG,IA),(IREG(2),IB)
      DATA INAME/2HFM,2HGR,2H  /,IPRM1/2HPA/,IPRM2/2HCK/,IPRM3/2H  /
       .
      REG = REXEC(ISTAT,10,INAME,IPRM1,IPRM2,IPRM3,0,6)
      IF (ISTAT.............
      IF (IA................
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REXUS"
    BEGIN
          .
  INTEGER INAME[1:3] := "FM","GR","  " ;
  INTEGER IA,IB,ISTAT ;
          .
     REAL
PROCEDURE REXEC(ISTAT,ICODE,INAME,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5) ;
    VALUE ICODE,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
  INTEGER ISTAT,ICODE,INAME,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
    CODE ;
PROCEDURE SEPAR(P1,P2,P3) ;
    VALUE P1 ;
     REAL P1 ;
  INTEGER P2,P3 ;
    CODE ;
          .
          SEPAR(REXEC(ISTAT,10,INAME[1],"PA","CK","  ",0,6),IA,IB) ;
          IF ISTAT.............. ;
          IF IA................. ;
          .
       END$
```

PTPOF

PURPOSE

Enable operator attention processing after completion of remote communication.

```
EXT     PTPOF
  .
  .
  .
JSB     PTPOF
DEF     * + 1
```

COMMENTS

This call should be made after program-to-program communication is complete and before program exit. See PTPON call.

SEQUENCE OF OPERATIONS

1. Clear $BUSY flag.
2. Return.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING FORTRAN AND ALGOL EXAMPLES WILL DEMONSTRATE THE PROPER SEQUENCE OF THE USE OF PTPON AND PTPOF.

----------FORTRAN----------

```
FTN,L
      PROGRAM REMOTE
        .
      CALL PTPON
        .
C     BEGIN REMOTE COMMUNICATIONS
        .
C     END REMOTE COMMUNICATIONS
        .
      CALL PTPOF
        .
      END
        .
      END$
```

----------ALGOL----------

```
4PAL,L,"REMOT"
    BEGIN
          .
PROCEDURE PTPOF ;
     CODE ;
PROCEDURE PTPON ;
     CODE ;
          .
          PTPON ; & BEGIN REMOTE COMMUNICATIONS
          .
```

```
COMMENT COMMUNICATIONS ENDED ;
          .
          PTPOF ;
          .
     END$
```

PTPON

PURPOSE

Lock out operator attention processing during remote communication. Signals program-to-program communication is in progress.

```
EXT     PTPON
.
.
.
JSB     PTPON
DEF     * + 1
```

COMMENTS

If operator attention processing is not locked out during remote program-to-program communication, the central program will stay in an I/O suspend loop if the terminal program is aborted. For this reason, this call should be issued immediately before a remote program schedule call if program-to-program communication is to take place. After a request is received, and before data is transmitted or received, a call may be made to ESCON to see if the ESCAPE key has been pressed. If so, a STOP may be transmitted to the central computer causing the central program to be terminated. See sample program in Section VII.

SEQUENCE OF OPERATIONS

1. Set $BUSY flag.
2. Return.

FORTRAN & ALGOL EXAMPLES

(See PTPOF)

PUT DATA STREAM

PURPOSE

This call is used at the terminal to send a data buffer in response to a Receive Data call from central. The data buffer length at the terminal must be exactly the same as the data buffer length at central. See Terminal Sends Data program example in Section VII.

```
        EXT     #TAM
         .
         .
         .
(A) =   Ø for send data
    =   1 for send request (user programs must not use this option)
        JSB     #TAM    Transfer control to TAM
        DEF     RTN     Return address
        DEC     6       Function code
        DEF     x       Request or data buffer
        DEC     y       Buffer length (negative bytes)
RTN     return point    Continue execution
         .
         .
         .
x       BSS     n       Request or data buffer to be written.
```

SEQUENCE OF OPERATIONS

1. Check the number of parameters.
2. Format .IOC. call using the caller's buffer address and buffer length.
3. Set up the mode for .IOC. call and set up for transmit request or transmit data.
4. Issue .IOC. call.
5. Wait for completion.
6. Set the A-Register to minus one, or the appropriate #TAM error code.
7. Return.

FORTRAN & ALGOL

This call cannot be made using FORTRAN or ALGOL.

RAPOS

PURPOSE

This routine sets the address of the next record for any remote file except type ∅.

```
        EXT     RAPOS
         .
         .
         .
        JSB     RAPOS       Subroutine call
        DEF     RTN         Return address
        DEF     ISTAT       Communication status
        DEF     IERR        Operational status
        DEF     NAME        File name
        DEF     IREC        Record
        DEF     IRS         Optional
        DEF     IOFF        Optional
RTN     return point        Continue execution
         .
         .
         .
ISTAT   BSS     1           Communication status returned here.
IERR    BSS     1           Error code returned here.
NAME    ASC     3,name      File name.
IREC    DEC     a           Record number of next record.
IRS     DEC     b           Relative block address of next record.
IOFF    DEC     c           Block offset of next record.
```

COMMENTS

The optional parameters IRS and IOFF are not required if a type 1 or 2 file is being positioned. READF or WRITF are the preferred calls to position type 1 or 2 files. RAPOS will, however, function correctly to allow sequential access of any cartridge file from the specified record.

RAPOS allows random access of sequential files. The parameters needed for RAPOS are relative and are not affected by packing; they are obtained by using a RLOCF call.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY POSITION FILE 'NAME' TO THE NEXT RECORD BY ABSOLUTE POSITIONING. THESE EXAMPLES WILL FIRST OBTAIN THE CORRECT POSITIONING INFO FROM THE 'RLOCF' INTRINSIC, AND MERELY PASS THE VALUES FOR THE NEXT RECORD ON TO THE 'RAPOS' INTRINSIC .

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
```

```
      DATA NAME/2HNA,2HME,2H  /
       .
      CALL RLOCF(ISTAT,IERR,NAME,IREC,IRS,IOFF)
      IF (ISTAT.....
      IF (IERR.....
      CALL RAPOS(ISTAT,IERR,NAME,IREC,IRS,IOFF )
      IF (ISTAT.....
      IF (IERR.....
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
       .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER IREC,IRS,IOFF  :
       .
       .
PROCEDURE RLOCF(ISTAT,IERR,NAME,IREC,IRS,IOFF) ;
  INTEGER ISTAT,IERR,NAME,IREC,IRS,IOFF :
    CODE ;
PROCEDURE RAPOS(ISTAT,IERR,NAME,IREC,IRS,IOFF ) ;
  INTEGER ISTAT,IERR,NAME,IREC,IRS,IOFF  ;
    CODE ;
       .
       .
        RLOCF(ISTAT,IERR,NAME[1],IREC,IRS,IOFF) ;
        IF ISTAT.............;
        IF IERR..............;
        RAPOS(ISTAT,IERR,NAME[1],IREC,IRS,IOFF) ;
        IF ISTAT.............    ;
        IF IERR.............    ;
       .
      END$
```

RCLOS

PURPOSE

This routine closes a remote file and makes the file available to other callers. RCLOS also, optionally, truncates the file size.

```
        EXT     RCLOS
         .
         .
         .
        JSB     RCLOS       Subroutine call
        DEF     RTN         Return address
        DEF     ISTAT       Communication status
        DEF     IERR        Operational status
        DEF     NAME        File name
        DEF     IRX         Number of blocks, optional
RTN     return point        Continue execution
         .
         .
         .
ISTAT   BSS     1           Communication status returned here.
IERR    BSS     1           Error code returned here.
NAME    ASC     3,name      File name.
IRX     DEC     n           Truncate option:
                              +n = number of blocks to be deleted from the end of the
                                   file when it is closed.
                              -n = retain main file, delete extents.
                               n = Ø = standard close.
```

COMMENTS

The following conditions must be met for the parameter IRX to be recognized:

- The current position must be in the main file (not an extent).
- The number of blocks to be deleted must be less than or equal to the number of blocks in the file.

If the number of blocks to be deleted equals the number of blocks in the file, the file is purged.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY CLOSE THE FILE 'NAME' AND DELETE 10 BLOCKS <IRX>=(10) FROM THE FILE.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
        .
        .
```

```
      DATA NAME/2HNA,2HME,2H  /,IRX/10/
        .
        .
      CALL RCLOS(ISTAT,IERR,NAME,IRX)
      IF (ISTAT.....
      IF (IERR.....
        .
        .
      END
        .
        .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER IRX := 10 ;
          .
          .
PROCEDURE RCLOS(ISTAT,IERR,NAME,IRX) ;
  INTEGER ISTAT,IERR,NAME,IRX ;
     CODE ;
          .
          .
          RCLOS(ISTAT,IERR,NAME[1],IRX) ;
          IF ISTAT.............;
          IF IERR..............;
          .
          .
      END$
```

RCONT

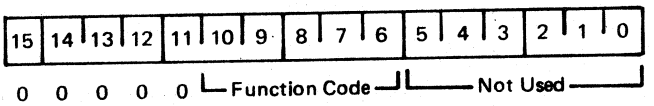
PURPOSE

This routine sends the standard RTE I/O Control request to type Ø (non-disc) remote files. It has no effect on other files.

```
        EXT     RCONT
         .
         .
         .
        JSB     RCONT   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT   Communication status
        DEF     IERR    Operational status
        DEF     NAME    File name
        DEF     ICON1
        DEF     ICON2   Optional
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
IERR    BSS     1       Error code or file type returned here.
NAME    ASC     3,name  File name.
ICON1   OCT     conwd   See Control Word below.
ICON2   DEC     n       Required for some functions (see Control Word below).
```

CONTROL WORD

The control word value (*conwd*) has one field that is ORed with the device logical unit number.



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

The following functions are defined for DVRØØ, DVRØ1 and DVRØ2:

2Ø	Enable auxiliary TTY-type device at central to schedule its program when any key is struck.
21	Disable auxiliary TTY-type device at central to inhibit scheduling of program.
22	Set time out – the optional third parameter is set as the new time out interval.
23	Ignore all further action requests until: a. The queue is empty. b. An input request is received. c. A restore control request is received.
24	Restore output processing (this request is usually not needed).

Function Code 11_8 (list output line spacing) requires the optional parameter ICON2. ICON2 must designate the number of lines to be spaced on the specified logical unit. A negative parameter specifies a page eject on a line printer.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY CONTROL THE FILE 'MT' TO REWIND-STANDBY(MAGNETIC TAPE UNIT) <ICON2>=(OCTAL 500) . IT IS ASSUMED THAT THE FILE 'MT' HAS BEEN PREVIOUSLY CREATED AS A TYPE Ø(ZERO) FILE WHICH POINTS TO THE MAGNETIC TAPE UNIT .

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
       .
      DATA NAME/2HMT,2H  ,2H  /,ICON1/B500/,ICON2/0/
       .
       .
      CALL RCONT(ISTAT,IERR,NAME,ICON1,ICON2)
      IF (ISTAT.....
      IF (IERR.....
       .
       .
      END
       .
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER ICON1 := @500 ;
  INTEGER ICON2 := 0 ;
          .
          .
PROCEDURE RCONT(ISTAT,IERR,NAME,ICON1,ICON2) ;
  INTEGER ISTAT,IERR,NAME,ICON1,ICON2 :
     CODE ;
          .
          .
          RCONT(ISTAT,IERR,NAME[1],ICON1,ICON2) ;
          IF ISTAT.....;
          IF IERR.....;
          .
          .
      END$
```

RCRET

PURPOSE

This routine creates the named file on the specified disc with the requested number of blocks. If the call is successfully completed, the file is left open to the caller.

```
        EXT     RCRET
         .
         .
         .
        JSB     RCRET   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT   Communication status
        DEF     IERR    Operational status
        DEF     NAME    File name
        DEF     ISIZE   File size (first word required)
        DEF     ITYPE   File Type
        DEF     IS      Optional security code
        DEF     ILU     Optional Cartridge Label or Logical Unit
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
IERR    BSS     1       Error code or number of sectors, if RCRET is successful,
                        returned here.
NAME    ASC     3,name  File name in ASCII.
ISIZE   DEC     a       Number of 128-word blocks in the file. If negative, the remainder
                        of the disc is assumed.
        DEC     b       Record length (for type 2 files only).
ITYPE   DEC     c       File types are defined in Comments.
IS      DEC     d       Security code.
ILU     DEC     e       Logical unit.
```

COMMENTS

Legal file types are:

1 = 128 word record length, random access.

2 = user selected record length, random access.

3 = (and greater) random record length, sequential access.

4 = source program.

5 = relocatable program.

6 = RTE load module.

7 = absolute program.

> 7 = user defined.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY CREATE A TYPE 2 <FIXED LENGTH> FILE CALLED 'NAME' WHOSE SECURITY CODE <IS> WILL BE 'DS' ON CARTRIDGE <ILU> # Ø .THIS FILE 'NAME'

WILL CONSIST OF 10 BLOCKS <ISIZE[1]>,AND EACH RECORD IN THE FILE
WILL BE 100 WORDS LONG <ISIZE[2]>.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3),ISIZE(2)
       .
       .
      DATA NAME/2HNA,2HME,2H  /,ISIZE/10,100/
      DATA ITYPE/2/,IS/2HDS/,ILU/0/
       .
       .
      CALL RCRET(ISTAT,IERR,NAME,ISIZE,ITYPE,IS,ILU)
      IF (ISTAT.........
      IF (IERR..........
       .
       .
      END
       .
       .
      END$
```

----------ALGOL----------

```
4PAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER ARRAY ISIZE[1:2] := 10,100 ;
  INTEGER ISTAT,IERR ;
  INTEGER ITYPE := 2 ;
  INTEGER IS := "DS" ;
  INTEGER ILU := 0 ;
          .
          .
PROCEDURE RCRET(ISTAT,IERR,NAME,ISIZE,ITYPE,IS,ILU) ;
  INTEGER ISTAT,IERR,NAME,ISIZE,ITYPE,IS,ILU ;
     CODE ;
          .
          .
          RCRET(ISTAT,IERR,NAME[1],ISIZE[1],ITYPE,IS,ILU) ;
          IF ISTAT.............    ;
          IF IERR..............    ;
          .
          .
      END$
```

REMOTE READ

PURPOSE

To transfer information to or from a remote I/O device.

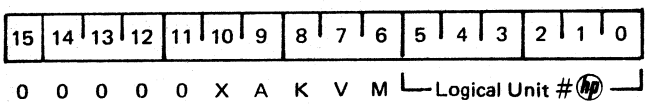
```
        EXT     REXEC
        .
        .
        .
        JSB     REXEC   Transfer control to remote RTE
        DEF     RTN     Point of return
        DEF     ISTAT   Communication status
        DEF     ICODE   Request code
        DEF     ICNWD   Control information
        DEF     IBUFR   Buffer location
        DEF     IBUFL   Buffer length
        DEF     IPRM1   Optional parameter
        DEF     IPRM2   Optional parameter
RTN     return point    Continue execution
        .
        .
        .
ISTAT   BSS     1       Communication status returned here.
ICODE   DEC     1       1 = READ.
ICNWD   OCT     conwd   conwd is described in Comments.
IBUFR   BSS     n       Buffer of n words (128 maximum).
IBUFL   DEC     n (or -2n)  Same n; words (+), or characters (-).
IPRM1   DEC     f       Decimal track number if disc transfer.
IPRM2   DEC     g       Decimal sector number if disc transfer.
```

COMMENTS

Parameters IPRM1 and IPRM2 are optional, except in the case of disc transfers. If the data transfer involves a disc, IPRM1 is the disc track number and IPRM2 is the disc sector number. In calls to other I/O devices these parameters may have other uses. For example, drive DVR77 (HP 2313A Subsystem) uses IPRM1 for the scanner channel number and IPRM2 for the instrument program word.

CONTROL WORD

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



Where:

X = 1 for moving head disc write with cyclic checking.

For paper tape devices, X indicates the honest mode.

If X, M and V are set, on input, absolute tape format is expected and handled.

If X and M are set on input, no leader is shipped and the specified number of words are read, or, output, no record terminator is punched (usually in 4 feed frames).

If X is set and M is not, on input; ASCII tape is read, no leader is shipped, bit 8 is stripped, but otherwise *all* characters are passed to the buffer. The *only* exception is the line feed which terminates the record.

On output, X rather than M, suppresses the carriage return line feed and forces non-recognition of any trailing left arrow, i.e., the left arrow is transmitted, but CRTLF is not.

A = 1 designates punching ASCII characters on the

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

teleprinter (M = ∅). ASCII is usually printed, but since it is sometimes desirable to punch ASCII tapes, this option is provided.

A = ∅ M, determines the mode of transfer.

Where:

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

V = 1, and M = 1 cause the length of punched tape input to be determined by the word count in the first non-zero read from the tape.

V = 0, and M = 1, the length of the punched tape input is determined by the buffer length specified in the EXEC call. M determines the mode of data transfer (if applicable).

M = 0 for ASCII

M = 1 for binary

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

End-of-operation information is transmitted to the program in the A- and B-Registers. This A-Register contains word 5 (status word) of the device EQT entry with bits 14 and 15 indicating the end-of-operation status as defined by the driver completion code. Bits 14 and 15 are set to zero if the device is up and set to one if the device is down. The B-Register contains a positive number which is the number of words or characters (depending upon program specification) actually transmitted.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL READ A RECORD FROM THE STANDARD INPUT DEVICE <ICNWD=5> AND OUTPUT THE SAME RECORD TO THE STANDARD LIST DEVICE <ICNWD=6>.

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
      DIMENSION IBUFR(128)
      DIMENSION IREG(2)
      EQUIVALENCE (IREG(1),REG,IA),(IREG(2),IB)
       .
      REG = REXEC(ISTAT,1,5,IBUFR,128)
      IF (ISTAT.............
      IF (IAND(IA,377B).........
       .
      REG = REXEC(ISTAT,2,6,IBUFR,IB)
      IF (ISTAT.............
      IF (IAND(IA,377B).....
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REXUS"
    BEGIN
       .
  INTEGER ARRAY IBUFR[1:128] ;
  INTEGER IA,IB ;
  INTEGER ISTAT ;
       .
```

```
     REAL
PROCEDURE REXEC(ISTAT,ICODE,ICNWD,IBUFR,IBUFL) ;
    VALUE ICODE,ICNWD,IBUFL ;
  INTEGER ISTAT,ICODE,ICNWD,IBUFR,IBUFL ;
     CODE ;
PROCEDURE SEPAR(P1,P2,P3) ;
    VALUE P1 ;
     REAL P1 ;
  INTEGER P2,P3 ;
     CODE ;
          .
          SEPAR(REXEC(ISTAT,1,5,IBUFR[1],128),IA,IB) ;
          IF ISTAT..................... ;
          IF IA AND @377 ............... ;
          .
          SEPAR(REXEC(ISTAT,2,6,IBUFR[1],IB),IA,IB) ;
          IF ISTAT..................... ;
          IF IA AND @377 .............  ;
          .
       END$
```

REMOTE WRITE

PURPOSE

To transfer information to or from a remote I/O device.

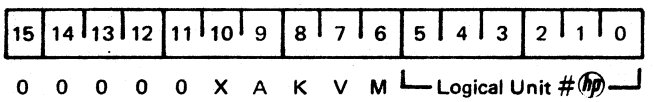
```
        EXT     REXEC
        .
        .
        .
        JSB     REXEC   Transfer control to remote RTE
        DEF     RTN     Point of return
        DEF     ISTAT   Communication status
        DEF     ICODE   Request code
        DEF     ICNWD   Control information
        DEF     IBUFR   Buffer location
        DEF     IBUFL   Buffer length
        DEF     IPRM1   Optional parameter
        DEF     IPRM2   Optional parameter
RTN     return point
        .
        .
        .
ISTAT   BSS     1       Communication status returned here.
ICODE   DEC     2       2 = WRITE.
ICNWD   OCT     conwd   conwd is described in Comments.
IBUFR   BSS     n       Buffer of n words (128 maximum).
IBUFL   DEC     n (or -2n)  Same n; words (+), or characters (-).
IPRM1   DEC     f       Decimal track number if disc transfer.
IPRM2   DEC     g       Decimal sector number if disc transfer.
```

COMMENTS

Parameters IPRM1 and IPRM2 are optional, except in the case of disc transfers. If the data transfer involves a disc, IPRM1 is the disc track number and IPRM2 is the disc sector number. In calls to other I/O devices these parameters may have other uses. For example, driver DVR77 (HP 2313A Subsystem), uses IPRM1 for the scanner channel number and IPRM2 for the instrument program word.

CONTROL WORD

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



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

Where:

X = 1 for moving head disc write with cyclic checking.

For paper tape devices, X indicates the honest mode.

If X, M and V are set, on input, absolute tape format is expected and handled.

If X and M are set on input, no leader is shipped and the specified number of words are read, or, output, no record terminator is punched (usually in 4 feed frames).

If X is set and M is not, on input; ASCII tape is read, no leader is shipped, bit 8 is stripped, but otherwise *all* characters are passed to the user buffer. The *only* exception is the line feed which terminates the record.

On output, X rather than M, suppresses the carriage return line feed and forces non-recognition of any trailing left arrow, i.e., the left

arrow is transmitted, but CRTLF is not.

A = ∅ M, determines the mode of transfer.

A = 1 designates punching ASCII characters on the teleprinter (M = ∅). ASCII is usually printed, but since it is sometimes desirable to punch ASCII tapes, this option is provided.

Where:

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

V = 1, and M = 1 cause the length of punched tape input to be determined by the word count in the first non-zero read from the tape.

V = 0, and M = 1, the length of the punched tape input is determined by the buffer length specified in the EXEC call. M determines the mode of data transfer (if applicable).

M = 0 for ASCII

M = 1 for binary

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

End-of-operation information is transmitted to the program in the A- and B-Registers. This A-Register contains word 5 (status word) of the device EQT entry with bits 14 and 15 indicating the end-of-operation status as defined by the driver completion code. Bits 14 and 15 are set to zero if the device is up and set to one if the device is down. The B-Register contains a positive number which is the number of words or characters (depending upon program specification) actually transmitted.

FORTRAN EXAMPLE

(See Remote Read)

ALGOL EXAMPLE

(See Remote Read)

RLOAD

PURPOSE

To load a program from the central computer into a terminal computer.

```
        EXT     RLOAD
         .
         .
         .
        JSB     RLOAD   Transfer control to TCE/3
        DEF     RTN     Point of return
        DEF     ISTAT   Communication status returned here (see Comments)
        DEF     IERR    Operational status
        DEF     IPROG   Name of program to load
        DEF     IADR    Starting address returned
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Status code returned here (see Comments).
IERR    BSS     1       Error code returned here (see Comments).
IPROG   ASC     3,xxxxx ASCII program name.
IADR    BSS     1       If the starting address is not available, a ∅ is returned in
                        this item.
```

COMMENTS

Legal values for the codes returned in STATUS for this call are:

- 5 = RFA busy
- 4 = Not used
- 3 = Not enough parameters
- 2 = Communications line down
- 1 = Illegal request code
- ∅ = Request being processed
- -1 = Request completed, no errors
- -2 = Not used
- -3 = I/O error
- -4 = Error in parameters
- -5 = Improper sequence of commands

Legal error codes returned in IERR for this call are:

Possible loading status returns are:

- ∅ = No error
- -1 = Open error on program name (file probably does not exist on central disc)
- -2 = File Read error at remote
- -3 = Transmission error for data record
- -4 = Transmission error, occurred when PROGL sent termination status to TCE/3
- -5 = File close error at remote
- -6 = Start address not available (no default address provided in location 3)

SEQUENCE OF OPERATIONS

1. Get user parameters from the calling sequence.
2. Get LU of the terminal from a GETLU call to the central computer.
3. Make a remote EXEC call to schedule PROGL. Pass parameters for LU and program name from the RLOAD call.
4. If PROGL is busy, return to the caller with IERR = 1.
5. Read data sent to the terminal by PROGL and store it into absolute core locations (program loading operation).
6. When the storage operation is complete, pass any error

code to the user call.

7. Save contents of location 3 for a default start address.

8. Return to the caller.

FORTRAN & ALGOL EXAMPLES

THE FORTRAN AND ALGOL EXAMPLES WILL LOAD A PROGRAM FILE FROM CENTRAL'S FILE SYSTEM.

----------FORTRAN----------

```
FTN,L
      PROGRAM REMOT
      DIMENSION IPROG(3)
      DATA IPROG/2HIP,2HRO,2HG /
       .
      CALL RLOAD(ISTAT,IERR,IPROG,IADR)
      IF (ISTAT.........
      IF (IERR.........
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REMOT"
    BEGIN
        .
  INTEGER ARRAY IPROG[1:3] := "IP","RO","G " ;
  INTEGER ISTAT,IERR,IADR ;
        .
PROCEDURE RLOAD(ISTAT,IERR,IPROG,IADR) ;
  INTEGER ISTAT,IERR,IPROG,IADR ;
     CODE ;
         .
          RLOAD(ISTAT,IERR,IPROG[1],IADR) ;
          IF ISTAT..........     ;
          IF IERR.......     ;
         .
      END$
```

RLOCF

PURPOSE

This routine formats and returns location and status information from the remote DCB.

```
        EXT     RLOCF
         .
         .
         .
        JSB     RLOCF   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT   Communication status
        DEF     IERR    Operational status
        DEF     NAME    File name
        DEF     IREC    Next record number
        DEF     IRS     Relative block of next record (optional)
        DEF     IOFF    Block offset of next record (optional)
        DEF     JSEC    Number of sectors or extents in the main file (optional)
        DEF     JLU     Returned LU of file device (optional)
        DEF     JTY     Returned file type (optional)
        DEF     JREC    Returned record size (optional)
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
IERR    BSS     1       Error code returned here.
NAME    ASC     3,name  ASCII File name.
IREC    BSS     1       Next record number returned here.
IRS     BSS     1       IRS includes extent information (i.e., IRS = JSEC*2*
                        Extent number plus relative block in current extent).
                        IRS equals IREC for type 1 files. IRS is not set for a
                        type Ø file.
IOFF    BSS     1       Set to Ø for type 1 files. Not set for type Ø files. Legal
                        values for other file types are: Ø ≤IOFF <128.
JSEC    BSS     1       Not set for type Ø files.
JLU     BSS     1       Used for disc and non-disc.
JTY     BSS     1       Same file type as set in the DCB.
JREC    BSS     1       Same value as directory entry, except:
                          Type 1 files, set to 128 for first read/write access.
                          Type Ø files, JREC is the read/write code.
                          Bit 15 = 1 for read legal; bit Ø = 1 for write legal.
```

RLOCF

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY ACCESS THE DCB MAINTAINED IN CENTRAL FOR FILE 'NAME' TO OBTAIN INFO ABOUT THE NEXT RECORD <IREC>.UPON RETURN,<IREC> WILL CONTAIN THE NEXT RECORD NUMBER;<IRS> WILL CONTAIN THE RELATIVE BLOCK OF THE NEXT RECORD;<IOFF> WILL CONTAIN THE BLOCK OFFSET OF THE NEXT RECORD;<JSEC> WILL CONTAIN THE NUMBER OF SECTORS IN THE MAIN FILE; <JLU> WILL CONTAIN THE CARTRIDGE NUMBER FOR THE FILE;<JTY> WILL CONTAIN THE FILE TYPE;AND <JREC> WILL CONTAIN THE RECORD SIZE FOR A TYPE 2 FILE.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
       .
      DATA NAME/2HNA,2HME,2H  /
       .
       .
      CALL RLOCF(ISTAT,IERR,NAME,IREC,IRS,IOFF,JSEC,JLU,JTY,JREC)
      IF (ISTAT.........
      IF (IERR.........
       .
       .
      END
       .
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",Ø,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER IREC,IRS,IOFF,JSEC,JLU,JTY,JREC ;
          .
          .
PROCEDURE RLOCF(ISTAT,IERR,NAME,IREC,IRS,IOFF,JSEC,JLU,JTY,JREC) ;
  INTEGER ISTAT,IERR,NAME,IREC,IRS,IOFF,JSEC,JLU,JTY,JREC ;
     CODE ;
          .
          .
          RLOCF(ISTAT,IERR,NAME[1],IREC,IRS,IOFF,
                JSEC,JLU,JTY,JREC) ;
          IF ISTAT.............    ;
          IF IERR..............    ;
          .
          .
     END$
```

RMESG

PURPOSE

Send a message to the operator display device at the central computer.

```
        EXT     RMESG
         .
         .
         .
        JSB     RMESG     Transfer control to TCE/3
        DEF     RTN       Point of return
        DEF     ISTAT     Communication status
        DEF     MSGAD     Message address
        DEF     MSGL      Message length
RTN     return point      Continue execution
         .
         .
         .
ISTAT   BSS     1         Communication status returned here.
MSGAD   ASC     n, ASCII  Maximum length is 36 words.
                  message
MSGL    DEC     n         n≤36 (positive value).
```

COMMENTS

The message displayed at central will be preceded with an ID code that identifies to the central operator the terminal that sent the message.

The message format is:

"=T*nn*: ASCII message from terminal *n*."

Where:

nn = logical unit number of the sending terminal.

SEQUENCE OF OPERATIONS

1. Get user calling parameters.
2. Move user message to the output buffer.
3. Store the ID header in the output buffer.
4. Get the L.U. of the terminal via a GETLU call. Convert the L.U. to ASCII, and store it in the ID header.
5. Set up the total message length.
6. Make a remote EXEC call to write a record on the remote operator display device.
7. Return to caller.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL TRANSMIT A MESSAGE TO CENTRAL'S OPERATOR REQUESTING THAT THIS TERMINALS MAGNETIC TAPE BE HUNG FOR SUBSEQUENT ACCESS BY THIS TERMINAL.

----------FORTRAN----------

```
FTN,L
      PROGRAM REMOT
      DIMENSION MSGAD(36)
```

```
      DATA MSGAD/2HHA,2HNG,2H M,2HAG,2H T,2HAP,2HE ,2H#R,2HAC,2H21,
     X     2H59,2H7 ,2HFO,2HR ,2HTE,2HRM,2HIN,2HAL,2H A,2HCC,2HES,
     X     2HS /
       .
      CALL RMESG(ISTAT,MSGAD,22)
      IF (ISTAT.......
C     REMOTE STATUS CALL(SEE REXEC(13....))
       .
      END
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"REMOT"
   BEGIN
       .
  INTEGER ARRAY MSGAD[1:36] := "HA","NG"," M","AG"," T","AP","E ",
                               "#R","AC","21","59","7 ","FO","R ","TE",
                               "RM","IN","AL"," A","CC","ES","S " ;
  INTEGER ISTAT ;
       .
PROCEDURE RMESGG(ISTAT,MSGAD,MEGL) ;
    VALUE MSGL ;
  INTEGER ISTAT,MSGAD,MEGL ;
    CODE ;
        .
         RMESG(ISTAT,MSGAD[1],22) ;
         IF ISTAT..... ;
COMMENT PERFORM REMOTE STATUS (SEE REXEC(13, ....)) ;
       .
      END$
```

RNAME

PURPOSE

This routine closes the remote DCB (if open) and then renames the specified remote file.

```
        EXT     RNAME
         .
         .
         .
        JSB     RNAME   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT   Communication status
        DEF     IERR    Operational status
        DEF     NAME    Present file name
        DEF     NNAME   New file name
        DEF     IS      Optional security code
        DEF     ILU     Optional cartridge label or logical unit
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
IERR    BSS     1       Error code returned here.
NAME    ASC     3,name  The file specified by NAME must not be open to any
                        other program when this call is made.
NNAME   ASC     3,nuname File's new name.
IS      DEC     a       If NAME has a non-zero security, the proper security code
                        must be furnished.
ILU     DEC     b       If ILU is furnished, only that cartridge is searched for
                        name; otherwise, all mounted cartridges are searched.
                        If all cartridges are searched, only the first file found with
                        the given name will be changed.
```

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL DEMONSTRATE HOW TO REMOTELY RENAME THE FILE <NAME[1]> 'NAME' WITH SECURITY CODE <IS> 'DS'.THE EXAMPLE ALSO SPECIFIES THAT THE FILE IS TO BE RENAME'ED <NNAME> 'NNAME'.THE CARTRIDGE IS NOT KNOWN,SO THE PROGRAM SENDS THE FMP A Ø(ZERO),THUS RELYING ON THE FMP TO FIND THE FIRST FILE THAT HAS THE NAME 'NAME'.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3),NNAME(3)
       .
       .
```

RNAME


```
      DATA NAME/2HNA,2HME,2H  /,NNAME/2HNN,2HAM,2HE /
      DATA IS/2HDS/,ILU/0/
        .
        .
      CALL RNAME(ISTAT,IERR,NAME,NNAME,IS,ILU)
      IF (ISTAT.............
      IF (IERR.............
        .
        .
      END
        .
        .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER ARRAY NNAME[1:3] := "NN","AM","E " ;
  INTEGER IS := "DS" ;
  INTEGER ILU := 0 ;
          .
          .
PROCEDURE RNAME(ISTAT,IERR,NAME,NNAME,IS,ILU) ;
  INTEGER ISTAT,IERR,NAME,NNAME,IS,ILU ;
     CODE ;
          .
          .
          RNAME(ISTAT,IERR,NAME[1],NNAME[1],IS,ILU) ;
          IF ISTAT.............;
          IF IERR.............;
          .
          .
      END$
```

RNPGM

PURPOSE

To execute a loaded program at the central computer.

```
        EXT     RNPGM
         .
         .
         .
        JSB     RNPGM   Transfer control to TCE/3
        DEF     RTN     Point of return
        DEF     IADR    Optional starting address
        DEF     p1      Optional user parameters passed to the executing program
        DEF     p5
RTN     return point    Continue execution
         .
         .
         .
IADR    OCT     xxxxxx  If IADR is not supplied (user wishes to take default start address
                        contained in location 3), and location 3 is zero, control will re-
                        turn to the caller.
p1      BSS     1       The B-Register points to this parameter string when control is
 .                      passed to the loaded program.
 .
 .
p5      BSS     1
```

COMMENTS

If the starting address is not known and optional parameters are desired, IADR is set to 2.

SEQUENCE OF OPERATIONS

1. Set B-Register to the address of the first optional parameter. If no options are specified, the parameters are zero.
2. If a start address is specified in the call, a jump is executed to that address.
3. If no start address is specified, a jump is executed to the default start address.
4. If neither a start address nor a default are specified, return to the caller.

FORTRAN & ALGOL EXAMPLES

THE FORTRAN AND ALGOL EXAMPLES BELOW WILL CAUSE A PREVIOUSLY LOADED TERMINAL PROGRAM TO BE EXECUTED STARTING AT THE ADDRESS 'IADR' SUPPLIED BY THE 'RLOAD' INTRINSIC .

----------FORTRAN----------

```
FTN,L
      PROGRAM REMOT
       .
      CALL RNPGM(IADR)
       .
      END
       .
      END$
```

RNPGM

----------ALGOL----------

```
HPAL,L,"REMOT"
    BEGIN
      •
PROCEDURE RNPGM(IADR,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5) ;
    VALUE IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
  INTEGER IADR,IPRM1,IPRM2,IPRM3,IPRM4,IPRM5 ;
    CODE ;
        •
        RNPGM(IADR,0,0,0,0,0) ;
        •
     END$
```

ROPEN

PURPOSE

This routine closes the remote DCB (if open), and then opens the named file.

```
        EXT     ROPEN
        .
        .
        .
        JSB     ROPEN     Subroutine call
        DEF     RTN       Return address
        DEF     ISTAT     Communication status
        DEF     IERR      Operational status
        DEF     NAME      File name
        DEF     IOP       Optional open parameters (see Comments)
        DEF     IS        Optional security code
        DEF     ILU       Optional Cartridge Label or Logical Unit
RTN     return point      Continue execution
        .
        .
        .
ISTAT   BSS     1         Communication status returned here.
IERR    BSS     1         If successful file type; otherwise, error code returned here.
NAME    ASC     3,name    File name.
IOP     OCT     a         Open option (see Comments).
IS      DEC     b         Security code.
ILU     DEC     c         Cartridge label.
```

COMMENTS

The IOP (open option) parameter is defined as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	X	A	K	V	M	0	0	F	T	U	E

Where:

E = ∅ for exclusive open (open to all terminals, but not to central programs).

E = 1 for non-exclusive open (open to all terminals and to central programs).

U = ∅ for non-update open.

U = 1 for update open.

T = ∅ means file type defined in directory.

T = 1 means force to file type 1 (extents are not addressable).

F = ∅ uses function code defined in directory.

F = 1 uses function code defined in bits 6 - 10 of this word for type ∅ file only.

M = ∅ for ASCII.

M = 1 for binary.

V = 1, and M = 1, causes the length of punched tape input to be determined by the word count in the first non-zero character read from the tape.

V = 0, and M = 1, the length of the punched tape input is determined by the buffer length specified in the call. M determines the mode of the data transfer (if applicable).

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

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

X is used only for type ∅ files.

When paper tape devices are used, "X", in combination with "M" and "V" will indicate an

honesty mode.

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

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

- **UPDATE OPEN.** Update implies that a block is read before it is modified so that existing records within the block will not be destroyed. Type 1 files are inherently update files. In type 1 and 2 files the end-of-file is the last word of the last block in the file. However, in other type files it is a special word (-1) for the length of a record. In order to preserve data in a type 2 file, it should be opened for update whenever any non-sequential accesses are to be performed, and the file is to be modified. That is to say, a type 2 file should be modified in non-update mode only when originally writing the file, or adding to the end of the file; and then only if it is to be written sequentially. Update/non-update has no effect on reading or positioning.

  In type 3 and above files, an end-of-file mark is written after each record that is written in the non-update fashion. That is, replace the end-of-file with the new record, followed by an end-of-file.

- **EXCLUSIVE OPEN.** In order to prevent central and terminal programs from destructively interferring with each other, a file may be opened either exclusively or non-exclusively. If one terminal opens a file exclusively, no other terminal or central may open the same file.

- **NON-EXCLUSIVE OPEN.** A non-exclusive open will allow central and terminal programs to open the same file. A non-exclusive open will not be granted if the file is already opened exclusively.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL DEMONSTRATE HOW TO REMOTELY OPEN THE FILE <NAME[1]> 'NAME' WITH SECURITY CODE <IS 'DS'.THE EXAMPLE ALSO SPECIFIES THAT THE FILE IS TO BE OPEN'ED EXCLUSIVELY.THE CARTRIDGE IS NOT KNOWN,SO THE PROGRAM SENDS THE FMP A Ø(ZERO),THUS RELYING ON THE FMP TO FIND THE FIRST FILE THAT HAS THE NAME 'NAME'.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
       .
      DATA NAME/2HNA,2HME,2H  /,IOP/Ø/,IS/2HDS/,ILU/Ø/
       .
       .
      CALL ROPEN(ISTAT,IERR,NAME,IOP,IS,ILU)
      IF (ISTAT.............
      IF (IERR.............
       .
       .
      END
       .
       .
      END$
```

ROPEN

----------ALGOL----------

```
HPAL,L,"RFA'R",Ø,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER IOP := Ø ;
  INTEGER IS := "DS" ;
  INTEGER ILU := Ø ;
          .
          .
PROCEDURE ROPEN(ISTAT,IERR,NAME,IOP,IS,ILU) ;
  INTEGER ISTAT,IERR,NAME,IOP,IS,ILU ;
     CODE ;
          .
          .
          ROPEN(ISTAT,IERR,NAME[1],IOP,IS,ILU) ;
          IF ISTAT.............;
          IF IERR.............;
          .
          .
      END$
```

PURPOSE

This routine allows the next remote read or write to access a given record in any remote file type.

```
        EXT     RPOSN
         .
         .
         .
        JSB     RPOSN     Subroutine call
        DEF     RTN       Return address
        DEF     ISTAT     Communication status
        DEF     IERR      Operational status
        DEF     NAME      File name
        DEF     NP        Number of records to skip
        DEF     IR        Optional absolute/Relative Counter for NUR
RTN     return point      Continue execution
         .
         .
         .
ISTAT   BSS     1         Communication status returned here.
IERR    BSS     1         Error code returned here.
NAME    ASC     3,name    File name.
NP      DEC     a         The NP (new record) parameter is defined as follows:
                            NP > ∅  Forward positioning, skip NUR records.
                            NP < ∅  Backward positioning, skip NUR records.
                            NP = ∅  No operation.
                          If a type ∅ file is involved, the motion requested must be legal
                          for the device.
IR      DEC     b         The IR (absolute vs. relative) parameter is defined as follows:
                            IR = ∅ or not present  The new record indicated by NUR is
                                                   relative to the present position in the file.
                            IR ≠ ∅  The new record indicated by NUR is the absolute
                                    number starting from the first record in the
                                    file (record number 1). Note that NUR must
                                    be > ∅.
```

COMMENTS

Forward positioning on a type ∅ file is accomplished by reading records until one less than the desired record is read, or an EOF is read. In all cases, EOF terminates positioning.

Type 3 and above files are treated as magnetic tape files. To be correct, a backspace should be issued after an EOF is read, and before continuing to write. This action simulates a magnetic tape, that is, the EOF is spaced over. It is also legal to continue without backspacing.

RPOSN

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY POSITION THE FILE 'NAME' TO THE PREVIOUS RECORD,RELATIVE <IR>=(Ø) POSITION <NP>=(-2) .

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
       .
      DATA NAME/2HNA,2HME,2H  /,NP/-2/,IR/Ø/
       .
       .
      CALL RPOSN(ISTAT,IERR,NAME,NP,IR)
      IF (ISTAT.....
      IF (IERR.....
       .
       .
      END
       .
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",Ø,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER NP := -1 ;
  INTEGER IR := Ø ;
          .
          .
PROCEDURE RPOSN(ISTAT,IERR,NAME,NP,IR) ;
  INTEGER ISTAT,IERR,NAME,NP,IR ;
     CODE ;
          .
          .
          RPOSN(ISTAT,IERR,NAME[1],NP,IR) ;
          IF ISTAT.............;
          IF IERR..............;
          .
          .
      END$
```

RPURG

PURPOSE

This routine closes the remote DCB (if open), and then purges the named remote file.

```
        EXT     RPURG
         .
         .
         .
        JSB     RPURG   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT   Communication status
        DEF     IERR    Operational status
        DEF     NAME    File to be purged
        DEF     IS      Optional security code
        DEF     ILU     Optional Cartridge Label or Logical Unit
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
IERR    BSS     1       Error code returned here.
NAME    ASC     3,name  Name of file to be purged.
IS      DEC     a       Security code.
ILU     DEC     b       Cartridge Label.
```

COMMENTS

The file must not be open to any other program when this call is made.

This routine will not purge type Ø (non-disc) files.

If the file being purged is the last file on the disc (excluding a type 6 file), all area from the last unpurged file up to and including the purged file, is returned to the system. If a purged file is not the last file, then its area may be recovered only by packing, or by purging all files after it in the directory. Type 6 file area may only be recovered by packing.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY PURGE THE FILE 'NAME' WITH SECURITY CODE <IS> 'DS' ON CARTRIDGE <ILU> # Ø .

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
       .
      DATA NAME/2HNA,2HME,2H  /,IS/2HDS/,ILU/Ø/
       .
       .
      CALL RPURG(ISTAT,IERR,NAME,IS,ILU)
      IF (ISTAT.....
```

RPURG


```
      IF (IERR.....
       .
       .
      END
       .
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER IS := "DS" ;
  INTEGER ILU := 0 ;
          .
          .
PROCEDURE RPURG(ISTAT,IERR,NAME,IS,ILU) ;
  INTEGER ISTAT,IERR,NAME,IS,ILU ;
    CODE ;
          .
          .
          RPURG(ISTAT,IERR,NAME[1],IS,ILU) ;
          IF ISTAT.............;
          IF IERR..............;
          .
          .
      END$
```

RREAD

PURPOSE

This routine reads a record from the remote file currently open to the remote DCB, into the user buffer.

```
        EXT     RREAD
          .
          .
          .
        JSB     RREAD     Subroutine call
        DEF     RTN       Return address
        DEF     ISTAT     Communication status
        DEF     IERR      Operational status
        DEF     NAME      Name of file
        DEF     IBUF      Data buffer
        DEF     IL        Required read request length
        DEF     L         Optional actual length parameter (returned)
        DEF     N         Optional record number to read
RTN     return point      Continue execution
          .
          .
          .
ISTAT   BSS     1         Communication status returned here.
IERR    BSS     1         Error code returned here.
NAME    ASC     3,name    Name of remote file to be read.
IBUF    BSS     n
IL      DEC     a         Legal values are >0 and ≤128.512
L       BSS     b         Contains the actual number of words transferred
                          into IBUF.
N       DEC     c         Used only for record types 1 and 2 (see Comments).
```

COMMENTS

The Request Length (IL) parameter is used differently by different file types:

- FILE TYPE Ø. The RREAD routine transfers up to IL words. If the record length is smaller than IL, only the record length is transferred. If the record is smaller, equal or greater than IL, no error is indicated.
- FILE TYPE 1. RREAD reads exactly IL words. Since the record length is fixed at 128 words, IL can be used to transfer less than a full record or more than one record. All records specified by IL, however, must be within the same file.
- FILE TYPE >1. RREAD transfers up to IL words. If the record length is smaller than IL, only the record length is transferred. No error is indicated whether the record is the same size, smaller, or larger than IL.

The actual length (L) parameter can never exceed IL. If L = IL, a check should be made to see if the record being transferred is too long (unless type 1 file is being read). If L = - 1, an end-of-file has been reached.

If the random access record number (N) is equal to zero or is not set, the remote file transfer starts at the current record number. If N is positive, the transfer starts at the record number indicated by N. If N is negative, the transfer starts at the current record plus N. Therefore:

File Type	Meaning
Ø	Not used.
1	N defines the first record transferred, and IL defines the number of records through the length in words.
2	N defines the record to be transferred.
3 and above	Not used.

There are three levels of EOF associated with file types that are defined below:

File Type	Meaning
Ø	When an EOF is read, parameter L is set to -1 with no error. Accesses may continue beyond the EOF.
1 and 2	When an EOF is read, parameter IERR is set to -12 indicating an error condition. No accesses beyond the EOF can be made.
3 and above	The first EOF read causes parameter L to be set to -1 with no error. A read access beyond this EOF causes an error condition (IERR = -12). A write access may continue beyond this EOF with no error.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY READ THE BUFFER <IBUF> OF LENGTH <IL> ON TO THE NEXT AVAILABLE RECORD OF FILE <NAME[1]> 'NAME' .THE TRANSMISSION LOG IS RETURNED IN <L>.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3),IBUF(100)
       .    .
       .
      DATA NAME/2HNA,2HME,2H  /
      DATA IL/100/,N/0/
       .
       .
      CALL RREAD(ISTAT,IERR,NAME,IBUF,IL,L,N)
      IF (ISTAT.........
      IF (IERR..........
      IF (L.............
       .
       .
      END
       .
       .
      END$
```

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","  " ;
  INTEGER ARRAY IBUF[1:100] ;
  INTEGER IL := 100 ;
  INTEGER L,N := 0 ;
          .
          .
PROCEDURE RREAD(ISTAT,IERR,NAME,IBUF,IL,L,N) ;
  INTEGER ISTAT,IERR,NAME,IBUF,IL,L,N ;
    CODE ;
          .
          .
```

RREAD


```
    RREAD(ISTAT,IERR,NAME[ ],IBUF[ ],IL,L,N) ;
    IF ISTAT.............    ;
    IF IERR..............    ;
    IF L.................;
  .
  .
END$
```

RSTAT

PURPOSE

This routine returns information on all cartridge labels in the remote system.

```
        EXT     RSTAT
         .
         .
         .
        JSB     RSTAT     Subroutine call
        DEF     RTN       Return address
        DEF     ISTAT     Communication status
        DEF     IBUF      Buffer
RTN     return point      Continue execution
         .
         .
         .
ISTAT   BSS     1         Communication status returned here.
IBUF    BSS     124       Buffer of 124 words.
```

COMMENTS

The cartridge status is contained in four-word increments.

Word	Meaning
∅	Logical unit number (first disc).
1	Last track for FMP.
2	Cartridge label.
3	Locking program's ID segment address, or ∅ if not locked.
4	Logical unit number (second disc).
.	
.	
.	
124	∅ standard terminator (not returned).

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL DETERMINE THE STATUS OF ALL THE REMOTE FMP'S CARTRIDGES.THESE STATII WILL BE STORED IN THE ARRAY <IBUF> .

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION IBUF(124)
       .
       .
      CALL RSTAT(ISTAT,IBUF)
      IF (ISTAT.....
       .
       .
      END
       .
       .
      END$
```

RSTAT

----------ALGOL----------

```
HPAL,L,"RFA'R",0,3,99
    BEGIN
          .
          .
  INTEGER ARRAY IBUF[1:124] ;
  INTEGER ISTAT ;
          .
          .
PROCEDURE RSTAT(ISTAT,IBUF) ;
  INTEGER ISTAT,IBUF ;
     CODE ;
          .
          .
           RSTAT(ISTAT,IBUF[1]) ;
           IF ISTAT.............;
          .
          .
      END$
```

RWIND

PURPOSE

This routine rewinds remote type Ø files, and sets remote disc files so that the next record is the first record in the file.

```
            EXT       RWIND
             .
             .
             .
            JSB       RWIND     Subroutine call
            DEF       RTN       Return address
            DEF       ISTAT     Communication status
            DEF       IERR      Operational status
            DEF       NAME      File name
RTN         return point        Continue execution
             .
             .
             .
ISTAT       BSS       1         Communication status returned here.
IERR        BSS       1         Error code returned here.
NAME        ASC       3,name    Name of file to be rewound.
```

FORTRAN & ALGO EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY REWIND THE FILE 'NAME' TO THE BEGINNING OF THE FILE.

----------FORTRAN----------

```
FTN,L
      PROGRAM RFA'R,3,99
      DIMENSION NAME(3)
       .
       .
      DATA NAME/2HNA,2HME,2H  /
       .
       .
      CALL RWIND(ISTAT,IERR,NAME)
      IF (ISTAT.....
      IF (IERR.....
       .
       .
      END
       .
       .
      END$
```

RLOIN

----------ALGOL----------

```
HPAL,L,"RFA'R",Ø,3,99
    BEGIN
          .
          .
  INTEGER ARRAY NAME[1:3] := "NA","ME","   " ;


          .
          .
PROCEDURE RWIND(ISTAT,IERR,NAME) ;
  INTEGER ISTAT,IERR,NAME ;
     CODE ;
          .
          .
           RWIND(ISTAT,IERR,NAME[1]) ;
           IF ISTAT.............;
           IF IERR..............;
          .
          .
      END$
```

RWRIT

PURPOSE

This routine writes a record on the remote file currently open to the remote DCB.

```
        EXT     RWRIT
         .
         .
         .
        JSB     RWRIT   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT   Communication status
        DEF     IERR    Operational status
        DEF     NAME    File name
        DEF     IBUF    Data buffer
        DEF     IL      Required write request length
        DEF     N       Optional random access record number
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
IERR    BSS     1       Error code returned here.
NAME    ASC     3,name  Name of remote file to be written on.
IBUF    BSS     a       Buffer containing data to be written.
IL      DEC     a       Legal values are: > Ø - ≤128 words (see Comments).
N       DEC     b       Record number where transfer is to start for record types
                        1 or 2 only (see Comments).
```

COMMENTS

The IL (write request length) parameter takes on different meanings depending on the file type:

File Type	Request Length (IL>Ø)
Ø	The routine will write exactly IL words.
1	Data is written in 128-word blocks. For example, IL is rounded up to 128 (1 block), if IL is between 1 and 127.
2	IL is ignored and the true file defined length is used.
>2	The routine will write exactly IL words.

If the random access record number (N) is equal to zero or is absent, the transfer starts at the current record number. If N is positive, the transfer starts at the record number indicated by N. If N is negative, the transfer starts at the current record plus N. Therefore:

File Type	Action
Ø	Not used.
1	N defines the first record to be written, and IL defines the number of records through the length in words.
2	N defines the record to be transferred.
3 and above	Not used.

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES IN FORTRAN AND ALGOL WILL REMOTELY WRITE THE BUFFER <IBUF> OF LENGTH <IL> ON TO THE NEXT AVAILABLE RECORD OF FILE <NAME[1]> 'NAME' .

STOP

PURPOSE

This call is used when a transmit request and transmit data call is received and for some reason the terminal program cannot handle it.

```
EXT     #TAM
 .
 .
 .
JSB     #TAM    Transfer control to TAM
DEF     * + 2   Return address
DEC     2       Function code
```

COMMENTS

A transmit STOP reply is returned to central to halt the data transfer at this time. The central program should check bit 4 of the EQT status word after completion of data transmission to see if the data was refused by the terminal.

SEQUENCE OF OPERATIONS

1. Format an .IOC. call to transmit a STOP reply.
2. Make the .IOC. call.
3. Wait for completion.
4. Set the A-Register to minus one, or the appropriate #TAM error code.
5. Return.

FORTRAN & ALGOL

This call cannot be made using FORTRAN or ALGOL.

TIME REQUEST

PURPOSE

Requests the current time recorded in the remote real-time clock.

```
        EXT     REXEC
         .
         .
         .
        JSB     REXEC   Transfer control to remote RTE
        DEF     RTN     Point of return
        DEF     ISTAT   Communication status
        DEF     ICODE   Request code
        DEF     ITIME   Time value array
RTN     return point    Continue execution
         .
         .
         .
ISTAT   BSS     1       Communication status returned here.
ICODE   DEC     11      Request code = 11.
ITIME   BSS     5       Time value array (5 words).
```

COMMENTS

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

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

FORTRAN & ALGOL EXAMPLES

THE FOLLOWING EXAMPLES WILL DETERMINE APPROXIMATELY HOW MUCH TIME IS REQUIRED FOR THE TERMINAL TO ACCESS CENTRAL'S CLOCK.

----------FORTRAN----------

```
FTN,L
      PROGRAM REXUS
      DIMENSION ITIME(5),ITIM1(5),ITIM2(5)
      DATA ITIME/100,60,60,24,365/
       .
      CALL REXEC(ISTAT,11,ITIM1)
      CALL REXEC(ISTAT,11,ITIM2)
      DO 10 I=1,4
      IF (ITIM1(I).LT.ITIM2(I))GO TO 10
      ITIM2(I)=ITIME(I)+ITIM2(I)
      ITIM1(I+1)=ITIM1(I+1)-1
10    ITIME(I)=ITIM2(I)-ITIM1(I)
       .
      END
       .
      END$
```

ALGOL

```
HPAL,L,"REXUS"
    BEGIN
  INTEGER ARRAY ITIM1[1:5],ITIM2[1:5],ITIME[1:5] := 100,60,60,24,365 ;
  INTEGER I,ISTAT ;

PROCEDURE REXEC(ISTAT,ICODE,ITIME) ;
    VALUE ICODE ;
  INTEGER ISTAT,ICODE,ITIME ;
    CODE ;

         REXEC(ISTAT,11,ITIM1[1]) :
         REXEC(ISTAT,11,ITIM2[1]) :
         FOR I:=1 TO 4
         DO BEGIN
            IF ITIM1[I] > ITIM2[I]
            THEN BEGIN
                 ITIM2[I] := ITIM2[I] + ITIME[I] ;
                 ITIM1[I+1] := ITIM1[I+1] - 1 ;
                 END ;
            ITIME[I] := ITIM2[I] - ITIM1[I] ;
            END ;

     END$
```

TCE/3 OPERATOR REQUESTS

Operator requests provide direct communication between the operator at the terminal computer and the 91701 operating software. A summary of the commands available to the TCE/3 operator is provided in Table 4-7 below.

Table 4-7. Summary of TCE/3 Operator Requests

TCE/3	Function
LOAD	Program load
RUN	Run program at given address
RUN *name*	Load & run program at terminal
RUNAT	Run program at octal address
ABORT	Program terminate
TELLOP	Message to central display
TIME	Terminal display time
ON *name*	Schedule program at central
CREATE *name*	Create file at central
CLOSE	Close central file
PURGE	Purge central file
DLIST	List directory
RENAME	Rename a file at central

Terminal operator requests are implemented for TCE/3 by breaking apart the syntax string input and building a remote file access, remote EXEC call to RFAIN, or program load request to TEXEC, as required. As part of this process, the message processor converts the appropriate ASCII parameters to binary and error checks them. It then performs the subroutine call to TEXEC or RFAIN.

Request	Resultant Program Call to TEXEC/RFAIN
ABORT	IDLE
CLOSE	RCLOS
CREATE	RCRET
DLIST	REXEC (schedule DLIST)
LOAD	REXEC (schedule PROGL)
ON	REXEC (schedule request)
PURGE	RPURG
RENAME	RNAME
RUN	RNPGM
TELLOP	REXEC (write)
TIME	REXEC (time)

When the remote call is complete, a colon prompt is displayed and control is returned to the terminal keyboard for input (except for RUN).

Keyboard input to TCE/3 is accomplished by pressing the ESCAPE key. The executive responds by returning a colon. It waits for an operator request followed by a carriage return. Null input is ignored and another prompt character is displayed. Illegal requests cause a SYNTAX ERROR to be displayed, followed by another prompt character.

Input requests are error checked for legal syntax. A blank or a comma delimits a command mnemonic. A comma delimits parameters. Imbedded blanks after the command mnemonic delimiters are ignored unless the parameter is a character string as in the TELLOP request. Octal parameters must be preceded with an "@" sign. If an error is made in entering the parameters, the backspace key or CONTROL, then H, are struck to delete the last character entered. Pressing RUBOUT deletes the entire line. Each request must be terminated with a carriage return. The operator requests are presented in alphabetical order and described in detail in the material that follows.

ABORT

PURPOSE

Terminate executing of the program at a terminal computer.

FORMAT

ABORT

COMMENTS

This request clears all pending remote calls, clears I/O and returns to the TCE/3 operator prompt loop.

No program name is specified in the request because it causes all programs in the user core area to be aborted.

EXAMPLE: A user program is executing at a terminal and the user decides to quit. He presses the ESCAPE key to get TCE/3's attention. When the colon is displayed, he types ABORT for a graceful exit.

SEQUENCE OF OPERATIONS

1. Clear $BUSY and $ESC flags.
2. Call .IOC. to clear the I/O on all logical units.
3. Call RFAIM (special entry point in RFAIN) to terminate all Remote File Access or Remote EXEC activity.
4. Call IDLE.

CLOSE

PURPOSE

Close a file at the central station computer.

FORMAT

CLOSE *name*

WHERE

name is an ASCII file name up to six characters in length. Trailing or leading blanks are not required.

COMMENTS

If the file is already closed when this request is issued, the message UNOPEN FILE is displayed at the terminal. The request is ignored.

SEQUENCE OF OPERATIONS

1. If a file name is not given, display SYNTAX ERROR and call IDLE.
2. Call RCLOS to close the file.
3. If an error occurs, display error message.
4. Call IDLE.

CREATE

PURPOSE

Create a file of specified size and type at the central computer.

FORMAT

CREATE,*name*,*#blocks*[,*record size*[,*type*[,*security*]]]

WHERE

#blocks Any positive non-zero value.

Record size Valid record sizes are 1 - 128 words for type 2 files (not required otherwise). If a record size of zero is input, a SYNTAX ERROR is returned.

Type The file type can be any of the seven listed below. The default value is 3.

- 1 = Fixed length random access – 128 word or multiples of 128 words
- 2 = Fixed length random access – any length
- 3 = Sequential access, random length, general
- 4 = Sequential access, random length – source
- 5 = Sequential access, random length – relocatable
- 6 = RTE load module
- 7 = Sequential access, random length – absolute

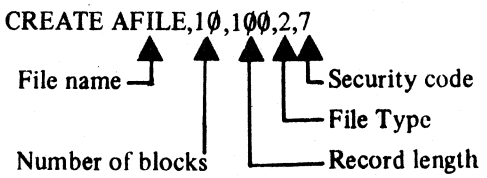
Security The security code is required to write on a file created with a non-zero code. Default value is zero.

- = Ø File is not protected and can be read or changed by any user.
- > Ø File is write protected, but can be read without a security code. If 2's complement of a positive code is used, the file will be opened for both read and write.
- < Ø File is protected and cannot be opened without the correct security code.

COMMENTS

The TCE/3 CREATE request results in an FMP CREAT program call at central, so that all restrictions for that call also apply to this request.

Example:



SEQUENCE OF OPERATIONS

1. If no file name or number of blocks are specified, SYNTAX ERROR is displayed and a call is made to IDLE.

2. If no record size parameter is given (used for type 2 files only), ignore it. If the record size is given, and it is negative, display SYNTAX ERROR.

3. If no file type is given, type 3 is set as a default.

4. If no security code is given, the default is set to zero.

5. All given parameters are converted to binary.

6. RCRET is called to create a file.

7. If an error occurs, the error message is displayed.

8. Call IDLE.

DLIST

PURPOSE

Selectively list the file directory.

FORMAT

DLIST [*filter*] [,*label*]

WHERE

filter is an optional filter specification. Each character specified in the filter must be matched by value and position with a character in the file name for the file to be listed.

label Cartridge number assigned when FMP is initialized at RTGEN.

COMMENTS

Security codes are ignored for this request.

label can be used to determine how much will be listed:

DLIST = List entire directory on all mounted discs.

DLIST,1 = List only cartridge number 1.

filter is used to determine what data is listed from each cartridge:

AB**** = List all file names starting with "AB."

CD = List names with 3rd, 4th characters equal to "CD."

*****3 = List names ending with "3."

P****1 = List names starting with "P" and ending with "1."

Note: A DLIST request executed at a terminal is not the same as a DL request executed at central.

SEQUENCE OF OPERATIONS

1. Call GETLU to obtain the logical unit at central for this terminal.
2. If a filter is given, a check is made for six characters. If the filter is not 6 characters, SYNTAX ERROR is displayed and IDLE is called.
3. If a cartridge label is given, it is converted to binary.
4. The $BUSY flag is set.

5. REXEC is called (ICODE = 10) to schedule DLIST at central. The parameters passed are the logical unit number, the filter (using 3 of the 5 optional parameters), and the cartridge label.

6. If DLIST is not dormant when the call is made at central, the schedule call is repeated until it becomes dormant. STANDBY is displayed at the first non-dormant return.

7. Program-to-program communication begins between TCE/3 at the terminal and program DLIST at central:

   a. The terminal calls #TAM to receive a two-word request from central. Completion occurs when DLIST at central begins execution at central and sends the request. This request tells TCE/3 if a data buffer is to follow and how many words it will contain.

   b. If no data buffer follows (DLIST at central is finished), $BUSY is cleared and IDLE is called.

   c. If a data buffer follows, #TAM is called to read the data. The DLIST program at central formatted an ASCII print line from data in the disc directory, and it is this data that is transmitted.

   d. The received print line is displayed on the terminal TTY.

   e. Steps a, b, c, and d are repeated until finished.

8. $BUSY flag is cleared.

9. IDLE is called.

SAMPLE OUTPUT FORMAT

```
CR=ØØØØØ1
  NAME      TYPE     #BLKS
 OVFIL     ØØØØØ1    ØØØØ5Ø
 RNAM      ØØØØØ7    ØØØØØ1
 PREC      ØØØØØ7    ØØØØØ1
 DOPEN     ØØØØØ7    ØØØØØ1
 DCLOS     ØØØØØ7    ØØØØØ1
 DPURG     ØØØØØ7    ØØØØØ1
 DWRIT     ØØØØØ7    ØØØØØ1
 DREAD     ØØØØØ7    ØØØØØ1
 DCRET     ØØØØØ7    ØØØØØ1
 DTIME     ØØØØØ7    ØØØØØ1
 DMSAG     ØØØØØ7    ØØØØØ1
 MONLK     ØØØØØ7    ØØØØØ1
 MON8      ØØØØØ7    ØØØØØ9
 RFAM/R    ØØØØØ5    ØØØØ41
 RFAMS     ØØØØØ5    ØØØØ41
 GENE      ØØØØØ3    ØØØØØ1
 JUNK1     ØØØØØ3    ØØØ5Ø
 JUNK      ØØØØØ1    ØØØØØ1
 SHA1      ØØØØØ5    ØØØØ41
 PROGØ3    ØØØØØ3    ØØØ13Ø
 JIM1      ØØØØØ3    ØØØØØ1
 A1        ØØØØØ3    ØØØØØ1
 A2        ØØØØØ3    ØØØØØ1
 A3        ØØØØØ3    ØØØØØ1
 A4        ØØØØØ3    ØØØØØ1
 A5        ØØØØØ3    ØØØØØ1
 A6        ØØØØØ3    ØØØØØ1
 A7        ØØØØØ3    ØØØØØ1
 A8        ØØØØØ3    ØØØØØ1
 RISFR     ØØØØØ3    ØØØØØ1
 RTSTR     ØØØØØ4    ØØØØØ1
:

:DLIST A*****
 CR=ØØØØØ1
  NAME      TYPE     #BLKS
 A1        ØØØØØ3    ØØØØØ1
 A2        ØØØØØ3    ØØØØØ1
 A3        ØØØØØ3    ØØØØØ1
 A4        ØØØØØ3    ØØØØØ1
 A5        ØØØØØ3    ØØØØØ1
 A6        ØØØØØ3    ØØØØØ1
 A7        ØØØØØ3    ØØØØØ1
 A8        ØØØØØ3    ØØØØØ1
:
```

LOAD

PURPOSE

Load named program at a terminal computer.

FORMAT

LOAD *name*

WHERE

name is the six ASCII character name of the program to be loaded.

COMMENTS

This request reads absolute binary format records from a remote disc file and stores them into terminal core, according to the load address contained in the record. When the request is complete, the terminal executive displays a prompt character (:) and waits for the next command to be entered.

The request is ignored if it contains a syntax error.

SEQUENCE OF OPERATIONS

1. SYNTAX ERROR is displayed if no program name is contained in the request.
2. RLOAD is called to load the program.
3. A WAIT is entered until completion.
4. A check is made for error codes sent from PROGL, or in behalf of PROGL. If PROGL was not dormant, STANDBY is output and another call is made to RLOAD.
5. If any errors occur, the appropriate error message is displayed.
6. IDLE is called.

ON name

PURPOSE

Schedule a program at central computer.

FORMAT

ON *name* [*p1, p2, p3, p4, p5*]

WHERE

name	is the name of the program to be scheduled. A maximum of six ASCII characters may be input.
p1 . . . p5	Optional parameters which may be passed to the central computer. Each parameter must be a positive integer with a value less than or equal to 32767_{10}.

COMMENTS

The address of the optional parameter string passed to the central computer is contained in the central computer B-Register when the program begins execution.

The program is placed in the schedule list and executed by priority and core availability. If all terminals input this request at the same time, central run sequence is determined by program priority. Program priority is established when the central program is RTGENed.

The difference between the ON request at the terminal and the ON used by RTE at central is that the terminal version does not have a "NOW" parameter. Both requests are used to schedule programs at central.

SEQUENCE OF OPERATIONS

1. If no *name* is contained in the request, SYNTAX ERROR is displayed and a call is made to IDLE.
2. Optional parameters (if any) are converted to binary.
3. REXEC is called (with ICODE = 10) to schedule the program and pass optional parameters. Parameters are zero if none were specified.
4. If errors occur, the appropriate error message is displayed. If the program to be scheduled is not dormant, the call to schedule the program is repeated until it is dormant.
5. IDLE is called.

PURGE

PURPOSE

Purge a file at the central computer.

FORMAT

PURGE *name* [,*security*]

WHERE

name from one to six ASCII characters

security the security code is required to delete a file created with a non-zero code.

PURGE AFILE, 7

COMMENTS

If the file being purged is the last file on the disc (excluding a type 6 file), all area from the last unpurged or type 6 file up to and including the purged file is returned to the system. If a purged file is not the last file, then its area may be recovered only by packing, or by purging all files after it is in the directory. Type 6 file area may only be recovered by packing.

SEQUENCE OF OPERATIONS

1. SYNTAX ERROR is displayed if no file name is given. IDLE is called.
2. The default value of zero is set if no security code is given.
3. RPURG is called to purge the file.
4. If errors occur, the appropriate error message is displayed.
5. IDLE is called.

RENAME

PURPOSE

Rename a file at the central computer.

FORMAT

RENAME *name, newname* [,*security*]

WHERE

name	one to six ASCII character file name.
newname	one to six ASCII characters.
security	default value is zero.

RENAME AFILE,BFILE,7

COMMENTS

The file specified by *name* must not be open to any other program when this request is issued. If *name* has a non-zero security, the proper security code must be included. All mounted cartridges are searched so that only the first file found with the given name will be changed.

It should be noted that if a file has a security code, and an attempt is made to either change the security code or delete it, using the RENAME request, an RMP error of -7 (bad file security code) will result.

SEQUENCE OF OPERATIONS

1. If either an old file name or a new file name are not given, SYNTAX ERROR is displayed and a call is made to IDLE.
2. A default value of zero is set if no security code is given.
3. RNAME is called to rename the file.
4. If errors occur, the appropriate message is displayed.
5. IDLE is called.

RUN

PURPOSE

Causes loaded program to be executed at a terminal computer, using the starting address loaded into location 3.

FORMAT

RUN [*,*p1, p2,. . . p5*]

WHERE

p1 . . . p5 Optional parameter string whose address is passed in the B-Register to the terminal program when the program begins execution.

RUN*,5,9,4ØØ, , 6

or

RUN

COMMENTS

The program to be run is determined by the address last loaded into location 3 when this request is issued.

If this request is issued while a program is running at the terminal, that program is restarted at its initial starting address (assuming no new address has been loaded into location 3).

SEQUENCE OF OPERATIONS

1. Optional parameters are converted to binary.
2. The B-Register is set to the address of the first binary parameter.
3. RNPGM is called to begin execution at the default starting address (location 3).

RUN name

PURPOSE

Causes the named program to be loaded and then executed at a terminal computer.

FORMAT

RUN *name* [,*p1, p2, p3, p4, p5*]

WHERE

name	is the ASCII name of the program to be run. This parameter may contain from one to six characters.
p1 . . . p5	an optional parameter string which may be passed to the terminal application program. Each parameter must be a positive integer $\leqslant 32767_{10}$.

RUN MYPROG,1,2,3,4,5

COMMENTS

The address of the binary parameter string is passed to the terminal program in the B-Register when the named program begins execution. If no optional parameters are specified, the B-Register points to a 5-word array of zeros.

SEQUENCE OF OPERATIONS

1. RLOAD is called to load the program and display the appropriate messages if errors occur. If PROGL is not dormant, STANDBY is displayed, and the RLOAD call is repeated until PROGL is dormant.

2. Optional parameters are converted to binary. The B-Register is set to the address of the first binary parameter.

3. RNPGM is called to begin execution at the default start address (location 3).

RUNAT

PURPOSE

Causes a program loaded at a terminal computer to be executed at a specified starting address.

FORMAT

RUNAT *nnn [,p1, p2, . . . p5]*

WHERE

nnn	starting address. If the starting address is in octal, it must be preceded with "@."
p1 . . . p5	Each parameter is a positive integer value less than or equal to 32767_{10}.

RUNAT@2000,1,2,3,4,5

COMMENTS

This request assumes that the program to be run has already been loaded as opposed to the RUN *name* request.

SEQUENCE OF OPERATIONS

1. Optional parameters are converted to binary.
2. The B-Register is set to the address of the first binary parameter.
3. RNPGM is called to begin execution at the given starting address.

TELLOP ASCII *message*

PURPOSE

To send a message to the central computer (operator) display device to perform a central RTE function (ready a peripheral, etc.).

FORMAT

TELLOP ASCII *message*

WHERE

The message that is displayed at the central computer is prefixed with "=T*nn*:" where *nn* = LU of the sending terminal. This prefix is added by the terminal executive and is not part of the message input.

The maximum length of a message is 72 characters, anything longer is truncated.

COMMENTS

This request is provided to allow communication from the terminal to the central operator.

EXAMPLE:

A terminal program requires output to the central line printer. If the line printer is not ready; the program can communicate via RMESG, or the operator can use TELLOP to tell the central operator to ready the line printer as shown below. The terminal in this example is LU12.

TERMINAL

TELLOP PLEASE READY LINE PRINTER

CENTRAL

=T12: PLEASE READY LINE PRINTER

SEQUENCE OF OPERATIONS

1. If the request does not contain a message, SYNTAX ERROR is displayed.
2. RMESG is called, with pointers to the message address and the word count.
3. The appropriate error message is displayed if an error occurs.
4. IDLE is called.

TIME

PURPOSE

Print time from central station clock on terminal computer display device.

FORMAT

TIME

COMMENTS

The response is a three-digit day of the year and the hours, minutes and seconds of time from a 24-hour clock.

SEQUENCE OF OPERATIONS

1. An REXEC call is made with ICODE = 11 to obtain time value from central real-time clock.
2. The appropriate error message is displayed if an error occurs.
3. The binary values for day/time are converted to ASCII.
4. The time is displayed on the local TTY.
5. IDLE is called.

TCE/3 ERROR MESSAGES

When the TCE/3 discovers an error in a remote call or request, it returns control to the user with ISTAT set to the appropriate error code. It is the user's responsibility to decide whether to ABORT to to take corrective action. The ISTAT error codes are:

5 = RFA busy
4 = Not used
3 = Not enough parameters
2 = Communications line down
1 = Illegal request code
Ø = Request being processed
-1 = Request completed, no errors
-2 = Not used
-3 = I/O error
-4 = Error in parameters
-5 = Improper sequence of commands (file not open to terminals)

REMOTE EXEC ERROR MESSAGES

The following conditions are returned to the terminal programs as STATUS = -4 (error in parameters):

Schedule Requests	INPUT/OUTPUT Requests
Missing parameter	Insufficient number of parameters
Illegal parameter	Illegal logical unit number
Referenced program cannot be scheduled	Logical unit not assigned
Referenced program is not defined	User buffer violates system boundaries
No resolution code in "TIMER" request	Illegal disc track or sector number in disc request
	Reference to protected track (not assigned to caller or using LGO before assigning Load-and-Go tracks)
	Disc transfer length exceeds tract boundary
	Overflow of Load-and-Go area

In addition to the error code, the following information is returned in the terminal A- and B-Registers:

READ/WRITE

A-Register = word 5 of the EQT Table (status)
B-Register = transmission log (number of words transmitted)
A-Register = Ø implies program scheduled

A-Register ≠ Ø implies program not scheduled, program status returned instead

Ø – Dormant
1 – Scheduled
2 – I/O suspend
3 – Not used
4 – Unavailable memory suspend
5 – Disc allocation suspend
6 – Operator suspend or programmed suspend

REMOTE FILE ACCESS ERROR MESSAGES

Error messages which result from operator requests to FMP are returned to the terminal operator in the TCE/3 message: "FMP ERR–XX". The XX values (-1 – -17) are listed below in the error Code column.

Error messages which result from program calls to FMP are passed back to the terminal computer programmer via the IERR parameter in the remote calling sequence. The values for IERR (-1 – -17) are listed below, along with the remote calls which may have been issued.

Table 4-8. Remote File Management Error Messages

ERROR CODE	ERROR DESCRIPTION	RCRET	RPURG	ROPEN	RCLOS	RREAD	RWRIT	RLOCF	RAPOS	RWIND	RPOSN	RNAME	RCONT
≥Ø	No error	X	X	X	X	X	X	X	X	X	X	X	X
-1	Disc down	X	X	X	X	X	X			X	X	X	
-2	Duplicate name	X										X	
-3	Backspace not legal										X		
-4	File too long or rec size error	X											
-5	Illegal record no., or attempt to read record not written, or illegal update length					X	X		X		X		
-6	Cartridge not found, or file not found, or no room	X	X	X	X		X						
-7	Invalid security code		X	X			X					X	
-8	File currently open: eight PGM, or exclusive, or lock rejected		X	X								X	
-9	Attempt to open type Ø as type 1 or use RAPOS or type Ø			X					X				
-10	Not enough parameters	X	X	X	X	X	X	X	X		X	X	
-11	DCB not open				X	X	X	X	X	X	X		X
-12	SOF or EOF read or sensed					X	X		X		X		X
-13	Cartridge locked	X	X	X								X	
-14	Directory full	X					X						
-15	Illegal name	X										X	
-16	Illegal type or size = Ø (RCRET)	X	X										
-17	Attempt to read or write or type Ø which does not support the operation.				X		X				X		

Table 4-9. TCE/3 Error Messages

Error Output	Meaning	Action
COM PROG ERR	Communications software malfunction	Used for debugging
COM LINE ERR	Communications hardware malfunction. TCE/3 may also indicate computer or communication cards not initialized or not running.	System aborts
SYNTAX ERROR	Illegal request	Verify syntax and retry
CAN'T LOCATE	Program to be scheduled at central or loaded at a terminal cannot be found on the central station disc.	
FMP ERR–XX	File manager error where XX = FMP error code (1 - 17). See RFA error messages for definitions.	See Table 4-8
UNOPEN FILE	Remote file not open to the terminal.	
PROGL ERROR	File read/write, or close error at a remote computer for a load, or run command checksum error.	
RFA BUSY	Either the local or the remote stack is full.	Retry until a slot becomes available
NO START ADR		Retry with RUNAT request
STANDBY	The central computer is busy at the moment, but will service this request as soon as free.	No additional input is necessary
MP VIOLATION	An attempt was made to overlay TCE/3 during a program load operation.	Operation aborted

TCE/3 HALTS

Fatal errors, hardware malfunctions, and software bugs cause a HALT 13B to occur. If such a halt occurs, the A-Register should be examined.

A-Register	Cause
1Ø77ØØ	The protected area of memory was left enabled. Disable and press RUN.
Equipment type code of last EQT	No EQT exists for D.65. TCE/3 must be re-generated.
13B	Unrecognizable reply from central for a RFA request. This is caused by a hardware malfunction or by the destruction of the TCE/3. B-Register contains the address of the reply buffer.
-1	There is a software bug at central. An error code sent by module ERR from central was one that should not occur. The B-Register contains the error code. Contact your HP Analyst.
Ø	A software bug in TCE/3. An error code generated by the module #TAM was one that should not occur. The B-Register contains the error code. Contact your HP Analyst.

TCE/3 INITIALIZATION

Load TCE/3 via TCE/1

1. Confirm TCE/1 in protected memory.
2. Press HALT.
3. Press SWITCH REGISTER.
4. Place TCE/3 program number in DISPLAY REGISTER (normally 3_8).

5. Press P-REGISTER.

6. Place $n7700_8$ in DISPLAY REGISTER

   where n = Ø - 4K, 1 - 8K, 2 - 12K, 3 - 16K, 5 - 24K, 7 - 32K

7. Press LOADER ENABLE, EXTERNAL PRESET, INTERNAL PRESET, and RUN.

8. The computer halts with 102077_8 in the DISPLAY REGISTER when TCE/3 is loaded correctly.

9. If the computer halts with 102011_8 in the DISPLAY REGISTER , TCE/3 is not loaded correctly. Reload following Steps 1 - 7 above.

Load TCE/3 via Paper Tape

1. Confirm BBL in protected memory.

2. Press HALT.

3. Press SWITCH REGISTER.

4. Clear DISPLAY REGISTER and place TCE/3 Binary Tape in Paper Tape Reader.

5. Press P-REGISTER.

6. Place $n7700_8$ in DISPLAY REGISTER

   where n = Ø - 4K, 1 - 8K, 2 - 12K, 3 - 16K, 5 - 24K, 7 - 32K

7. Press LOADER ENABLE, EXTERNAL PRESET, INTERNAL PRESET, and RUN.

8. The computer halts with 102077_8 in the DISPLAY REGISTER when the TCE/3 tape is completed.

9. If the computer halts with 102055_8 or 102011_8 in the DISPLAY REGISTER, a Paper Tape Read error has occurred. Retry following Steps 1 - 7 above.

Run TCE/3

1. Press P-REGISTER.

2. Set the DISPLAY REGISTER to the Starting Address of 2_8 (0/000/000/000/010).

3. Press INTERNAL PRESET.

4. Press EXTERNAL PRESET.

5. Press RUN.

Load TCE/3 from TCE/2 and Run

1. Confirm TCE/2 is in core.

2. When the prompt character (:) is received from TCE/2 input the following: RUN PROGØ3 (it is assumed that TCE/3 is called PROGØ3).

3. When the prompt (:) is returned, TCE/3 is running.

PART IV

CENTRAL SYSTEM OPERATION

INTRODUCTION

All of the functions normally found in the Real-Time Executive System and the File Management Package are available to the central programmer. A summary of these capabilities are provided under Program Calls below. Additional information can be found in the **Real-Time Executive Software Manual** and the **Real-Time Executive File Manager System Manual.** In addition to the RTE and FMP functions, the central programmer using 91701 software may communicate with a program running at a terminal using the program-to-program feature discussed in Section V of this manual.

Operator requests available to the Distributed System user at the central computer are also exactly those requests normally found in the Real-Time Executive system and the File Management Package interface, FMGR. A summary of these requests are provided in this section. The RTE and FMP manuals should be consulted for complete information.

A summary of RTE and FMP error codes are also provided with an indication of the reason for the error.

Programs may also be prepared at the central computer for execution at a terminal computer through the uses of the RTE Assembler and FORTRAN Compilers with the SXL Loader. Section III should be consulted for more information on this process.

PROGRAM CALLS

The summary of program calls available to the central programmer are divided by subsystem into RTE calls and FMP calls. Table 4-10 shows a summary of the RTE calls, and Table 4-11 provides a summary of the FMP calls, the functions they perform and the parameters required.

Table 4-10. Summary of RTE Calls

Call	Parameters		Function
DISC ALLOCATION	ICODE = 4	Allocate track to program	Request allocation of contiguous tracks for program.
	ITRAK	Number of contiguous tracks available	
	ISTRK	Starting track number returned	
	IDISC	Disc logical unit returned	
	ISECT	Number of 64-word sectors returned	
DISC RELEASE	ICODE = 5	Release program's tracks	Release some disc tracks assigned to to the program.
	ITRAK	Number of tracks to release	
	ISTRK	Starting track number of release returned	
	IDISC	Disc logical unit returned	
EXECUTION TIME	ICODE = 12		Schedule a program by time.
	IPROG	Program to be scheduled	
	IRESL	Resolution code	
	MTPLE	Execution multiple	
	IOFST	Units for initial offset	
	or		
	IHRS MINS ISECS MSECS	Define absolute start time	
GLOBAL DISC ALLOCATION	ICODE = 15	Allocate track globally	Request global allocation of contiguous tracks.
	ITRAK	Number of contiguous tracks desired	
	ISTRK	Starting track returned	
	IDISC	Disc logical unit returned	
	ISECT	Number of 64-word sectors returned	
GLOBAL DISC RELEASE	ICODE = 16	Release global tracks	Global release of some contiguous tracks.
	ITRAK	Number of tracks to release	
	ISTRK	Starting track number returned	
	IDISC	Disc logical unit returned	

(Continued)

Table 4-10. Summary of RTE Calls (Continued)

Call	Parameters		Function
I/O CONTROL	ICODE = 3 ICNWD IPRAM	 Control word Optional parameter required for some CONWD's	Carry out I/O functions, such as backspace, write end-of-file, rewind, etc.
I/O STATUS	ICODE = 13 ICNWD ISTA1 ISTA2	 Logical unit number Word 5 of EQT entry returned Optional parameter for word 4 of EQT	Request device status.
PROGRAM COMPLETION	ICODE = 6		Signal end of program.
PROGRAM SCHEDULE WITH WAIT	ICODE = 9 INAME *p1* . . . *p5*	Put program in waiting status Program name Optional parameters	Schedule a program for execution, and optionally, pass parameters to it.
PROGRAM SCHEDULE WITHOUT WAIT	ICODE = 10 INAME *p1* . . . *p5*	No wait Program name Optional parameters	Schedule a program for execution according to its priority.
PROGRAM SEGMENT LOAD	ICODE = 8 INAME	 Name of segment to load	Load a segment of calling program.
PROGRAM SUSPEND	ICODE = 7		Suspend calling program.
READ	ICODE = 1 ICNWD IBUFR IBUFL IPRM1 IPRM2	 Control word Buffer location Buffer length (words/characters) Optional parameter used for disc track in disc call Optional parameter used for disc sector in disc call	Transfer input.

(Continued)

Table 4-10. Summary of RTE Calls (Continued)

Call	Parameters		Function
TIME REQUEST	ICODE = 11		Request 24-hour time and day.
	ITIME	Time values: tens of milliseconds, seconds, minutes, hours, day returned	
WRITE	ICODE = 2		Transfer output.
	ICNWD	Control word	
	IBUFR	Buffer location	
	IBUFL	Buffer length (words/characters)	
	IPRM1	Optional parameter used for disc track in disc call	
	IPRM2	Optional parameter used for disc sector in disc call	

Table 4-11. Summary of FMP Calls

Call	Parameters		Function
APOSN	IREC	Record number of next record.	Position File to address of next record.
	IRB	Relative block address of next record.	
	IOFF	Block offset of next record.	
CLOSE	ITRUN	+*n* = number of blocks to be deleted from the end of the file when it is closed.	Close a DCB.
		−*n* = retain main file, delete extents.	
		n = Ø = standard close.	
CREAT	NAME	File name.	Create a file.
	ISIZE	Number of blocks in the file. If the value is negative, all of the disc is implied.	
		Record length (type 2 file only).	
	ITYPE	File type.	
	ISECU	Security code.	
	ICR	Cartridge Label.	
FCONT	ICON1	Control information	Send RTE control request to type Ø file.
	ICON2	Function code	
FSTAT	ISTAT	Buffer of 125 words.	Return cartridge status
LOCF	IREC	Next record number.	Return information on DCB.
	IRB	Relative block or next read.	
	FOFF	Block offset of next record.	
	JSEC	Number of sectors in the file.	
	JLU	File logical unit.	
	JTY	File type.	
NAMF	NAME	File's present name.	Rename a specified file.
	NNAME	File's new name.	
	ISECU	Security code.	
	ICR	Cartridge label.	
OPEN	NAME	File name.	Open specified file.
	IOPTN	Open options.	
	ISECU	Security code.	
	ICR	Cartridge label.	
POSNT	NUR	New record number.	Position the next access to a specific record.
	IR	Absolute/relative control parameter for NUR.	
PURGE	NAME	File name.	Purge a specified file.
	ISECU	Security code.	
	ICR	Cartridge Label.	

Table 4-11. Summary of FMP Calls (Continued)

Call	Parameters		Function
READF	IBUF	Data buffer.	Read a record from an open file.
	IL	Read request length.	
	LEN	Actual read length.	
	NUM	Record number to read.	
RWNDF	NO	Optional parameters.	Rewind a type Ø file or reset a disc file.
WRITF	IBUF	Data buffer.	Write a record to an open file.
	IL	Write request length.	
	NUM	Record number to write.	

OPERATOR REQUESTS

Operator requests input through the system input device are used to control the executing Real-Time Executive subsystem and RTE File Manager (FMP). The Distributed System operator at the central computer has full access to all functions available in both subsystems. Central computer operator requests can be used to:

a. Turn programs on and off
b. Suspend and restart programs
c. Examine status of programs and I/O devices
d. Declare I/O devices up or down
e. Copy files from one cartridge to another
f. Dump files
g. List file contents
h. Purge files
i. Create new files
j. Restore programs

Real-Time Executive Requests

The operator gains the attention of RTE by pressing any key on the console. When RTE responds with an asterisk (*), the operator types any operator request, consisting of a two-character request word and the appropriate parameters separated by commas. Two commas in succession mean that a parameter value is zero. Table 4-12 below provides a summary of all of the RTE operator requests, their functions, and required parameters.

Table 4-12. Summary of RTE Operator Requests

Request Word	Parameters		Function
DN	*n*	EQT entry number of I/O device to be set down	Declare an I/O device unavailable for use by the RTE system.
EQ	*n*	EQT entry number of the I/O device	Print EQT entry description and status of an I/O device.
EQ	*n* *p*	EQT entry number of the I/O device Delete/specify buffering	Change the automatic output buffering designation of an I/O device.
GO	*name* *p1* . . . *p5*	Name of an operator suspended program to be scheduled for execution List of parameters to be passed to the program	Reschedule a program suspended by SS operator request or suspend EXEC call.
IT	*name* *r* *mpt* *hr* *min* *s* *ms*	Name of the program to be run Resolution code 1 = tens of milliseconds 2 = seconds 3 = minutes 4 = hours Multiple execution interval value provides number for *r* units Hours Minutes Seconds Tens of m's } Set initial start time	Set automatic execution times for a program.
LG	*n*	Number of contiguous tracks to be released. If *r* = ∅, the allocated load-and-go area is released.	Allocate or release disc tracks for load-and-go operations.
LS	*p1* *p2*	Logical unit number of disc containing source file Starting track number of source file (decimal)	Designate disc logical unit number and starting track number of the existing source file.
LU	*n* *m* *p*	Logical unit number (decimal) EQT entry number to assign to *n*, or if zero, releases logical unit number Subchannel number to assign to *n*	Print or change logical unit number assignment. The print option is indicated if *m* and *p* values are absent.

(Continued)

Table 4-12. Summary of RTE Operator Requests (Continued)

Request Word	Parameters		Function
OF	*name* *p*	Name of program If $p = \emptyset$, terminates program and removes from time list. If $p > \emptyset$, $\neq 8$, terminates, removes from list and releases tracks. If $p = 8$, all of the above, plus removes from core RTE system.	Terminate program or remove background program.
ON	*name* *NOW* *p1* . . . *p5*	Name of program Schedules program immediately instead of by clock time Optional parameters passed to the program	Schedule program for execution.
PR	*name* *n*	Name of program New priority value	Change program priority.
RT	*name*	Name of program whose tracks are to be released	Release disc tracks assigned to program.
SS	*name*	Name of program to be suspended	Suspend program execution.
ST	*name*	Name of program whose status is to be printed	Request program status.
TI			Print time from real-time clock.
TM	*day* *h, min, s*	Three-digit day of the year The current time on a 24-hour clock	Set real-time clock.
TO	*n* *m*	EQT entry number of the I/O device Number of 10 *ms* intervals to be used as the time-out value	Print/change I/O device time-out parameter.
UP	*n*	EQT entry number of the device to be set up	Declare I/O device up.

FMP OPERATOR REQUESTS

Table 4-13 shows a summary of the File Management Package (FMP) operator requests available to the Distributed System operator at the central computer. Complete information for these requests may be found in the **Real-Time File Manager System Manual.**

The operator gains the attention of FMP by scheduling the software module FMGR with an ON FMGR operator request to RTE. FMGR is ready to receive requests when a system generated colon (:) is returned to the initial keyboard input device.

Each command is parsed by a central routine which accepts the conventions described below:

- The leading plus (+) sign in a numeric parameter is ignored. The number is assumed to be positive unless preceded by a minus (-) sign.
- A numeric parameter immediately followed by the letter "B" indicates that the parameter is octal.
- Each FMGR command from a device other than a TTY must be preceded by a colon (:). Entries from a TTY are prompted by a system generated colon – the operator must not enter a second colon.
- Each parameter is tested on the assumption that it is numeric. If the parameter is ASCII, the first six characters are saved. If less than six characters are entered, the parameter is padded with trailing blanks.
- Two commas or colons in a row mean a parameter is assigned its default value.
- Leading blanks and blanks on either side of a comma or semicolon are ignored.
- An EOF encountered on the input device causes a TR with no parameters to be output to the log device.
- All parameters are initially filled with zeros and flagged as not supplied.

Table 4-13. Summary of FMGR Operator Requests

Request Word	Parameters		Function
CL			List disc directory
CO	*label 1* *label 2*	Cartridge label or logical unit of present location Cartridge label or logical unit of new location	Copy all files on one disc to another disc
CR	*namr*	File name and parameters	Create a file
DC	*label*	Cartridge label or logical unit	Logically disconnect cartridge from FMP system
DL	*label* *master security code*	Cartridge label or logical unit Two-character FMP master security code from initialization	List file directory
DU	*namr1* *namr2* *record format* *EOF control* *file #* *#files*	Name of file or logical unit from which data is transferred Name of file or logical unit to which *namr1* is to be transferred Record format of *namr2* Saves or inhibits end-of-file marks on *namr2* Relative file number of *namr2* to be written in Number of files to be transferred from *namr1*	Transfer contents of one file to another file
EX			Terminate the file manager (FMGR)
IN	*master security code* *label 1* *label 2* *id* *1st trk* *#dir trks* *#sec/trk* *bad tracks*	Two-character FMP master security code from initialization Cartridge label or logical unit number New cartridge reference number Cartridge information label First track used on cartridge Number of directory tracks Number of 64-word sectors per track Bad track list	Initialize a disc
LI	*namr* *source* *Binary* *Directory*	Name of file or logical unit number ASCII source format Binary format Print directory entry, only	List file contents

(Continued)

Table 4-13. Summary of FMGR Operator Requests (Continued)

Request Word	Parameters		Function
LL	*namr*	Name of file or logical unit number	Change assignment of output list device
LO	*Lu*	Logical unit number of new log device	Change assignment of log device
MC	*Lu* *Last track*	Logical unit number of cartridge being mounted Last track on the cartridge available to FMP	Notify FMP cartridge has been mounted
MR	*namr*	Name of the file or logical unit number to be transferred	Move a relocatable binary file to load-and-go tracks
MS	*namr* *prog name* *IH*	Name of the file of logical unit number to be transferred Logical source (LS) tracks assigned to this program. Default; LS tracks assigned to EDIT. If supplied, the LS word on the base page is not set.	Move a source file to the logical source tracks
PK	*label*	Cartridge label or logical unit number to be packed	Pack disc
PU	*namr*	Name of file or logical unit to be purged	Purge a file
RN	*namr* *nuname*	Existing file name and parameters New name, unique to the cartridge	Rename a file
RP	*namr* *program*	Name of type 6 file to be restored Program name	Restore a program (file) saved by FMP so it can run in the RTE system
SA	*LS* *LG* *namr*	Logical source area Load-and-go area Name of file created by this command, or the logical unit number	Save the logical source or load-and-go area in file
SP	*namr*	Name of disc resident program	Place a disc resident program into a type 6 file
ST	*namr1* *namr2* *record format*	Name of file or logical unit from which data is transferred Name of file or logical unit to which data is being transferred ASCII, binary relocatable, binary, binary absolute, magnetic tape (ASCII), magnetic tape (SIO), magnetic tape (SIO binary relocatable), magnetic tape (SIO binary absolute) record format for *namr1*	Transfer records from a file or logical unit number to another file or logical unit number

(Continued)

Table 4-13. Summary of FMGR Operator Requests (Continued)

Request Word	Parameters		Function
ST (Cont'd)	*EOF control*	Save or inhibit EOF marks on *namr1*	
	file #	Relative position of file on *namr1* to be read	
	# files	Number of files to be transferred from *namr1*	
SV	*number*	New severity code number ∅ – Echo commands on log device 1 – Inhibit command echo 2 – Inhibit error messages	Change system log severity code
TR	*namr*	File name or logical unit number	Transfer control of FMGR to a file name or logical unit number
	-integer	Transfer back a number of *namr's*	
??	*number*	Error code number	Expand the last error message

ERROR MESSAGES

Error routines at the central computer perform a variety of checks on system operation and the validity of input messages are produced which inform the programmer or operator of the nature of the trouble when any of these checks fail. Some errors are forgivable and the system attempts to continue operation; others are disastrous and result in bypassing the remainder of the processing and aborting the system. These errors indicate, in general, failures in hardware or system bugs. In addition, error messages are produced by incorrect syntax, illegal parameters, etc. The tables which follow provide a summary of the error messages available to the programmer operator at the central computer. The RTE and FMP manuals should be consulted for further information.

RTE Error Messages

RTE errors shown in Table 4-14 cause a program to be terminated and any assigned disc tracks to be released. An error message is output on the operator console and RTE proceeds to execute the next program in the scheduled list.

Table 4-14. Summary of RTE Error Messages

Source	Message	Meaning
OPERATOR REQUESTS	OP CODE ERROR	Illegal operator request word
	NO SUCH PROG	The *name* given is not a main program in the system
	INPUT ERROR	A parameter is illegal
EXEC CALLS	*name* ABORTED	RTE aborted the program
	MP *name address* (*address* is the location that caused the violation).	Memory protect violation that is not an Exec Call
	RQ *name address* (*address* location that made the illegal call).	An Exec Call contains an illegal request code
SCHEDULE CALLS	SCØ1	Missing parameter
	SCØ2	Illegal parameter
	SCØ3 INT *name*	External interrupt attempts to schedule program already scheduled
	SCØ5	Program given is not defined
	SCØ6	No resolution code in Execution Time Exec call
DISC ALLOCATION CALLS	DRØ1	Insufficient number of parameters
(Continued)	DRØ2	Number of tracks is zero, >255, or <zero; illegal logical unit; or number of tracks to release is zero or negative

Table 4-14. Summary of RTE Error Messages (Continued)

Source	Message	Meaning
DISC ALLOCATION CALLS (Continued)	DRØ3	Attempt to release track assigned to another program
I/O CALLS	IOØ1	Not enough parameters
	IOØ2	Illegal Logical Unit
	IOØ3	Logical Unit not assigned
	IOØ4	Illegal user buffer
	IOØ5	Illegal disc track or sector
	IOØ6	Reference to a protected track or using load-and-go before assigning load-and-go tracks
	IOØ7	Driver rejected call
	IOØ8	Disc transfer longer than track
	IOØ9	Overflow of load-and-go area
EQUIPMENT MESSAGES	I/O ERR ET EQT *#n*	End-of-tape condition on device *#n*
	I/O ERR TO EQT *#n*	Device *#n* has timed-out
	I/O ERR PE EQT *#n*	Parity error in data transmission from device *#n*
	TR*nnnn* EQT *mm*, *upp s* (or *u*)	Disc transfer parity error
FORTRAN COMPILER ERROR MESSAGES	I/O ERR ET EQT *#n*	End of source tape
	IOØ6	FTN did not define load-and-go tracks
	IOØ9	FTN overflowed load-and-go tracks
ALGOL MESSAGES (Continued)	I/O ERR ET EQT *#n*	End of source tape encountered
	NO SOURCE	Source pointer not set for indicated input from disc

Table 4-14. Summary of RTE Error Messages (Continued)

Source	Message	Meaning
ALGOL MESSAGES (Continued)	IOØ6	HPAL did not define load-and-go tracks
	IOØ9	HPAL overflowed load-and-go tracks
ASSEMBLER ERROR MESSAGES	I/O ERR ET EQT #*n*	End of source tape encountered
	$ END ASMB CS	Error in assembler control statement
	$ END ASMB XEND	End-of-file condition occurred before END statement
	$ END ASMB NPRG	Source input for Logical Unit 2 (disc) requested, but no file declared
	IOØ6	Load-and-go tracks no defined
	IOØ9	ASMB overflowed load-and-go area
EDITOR ERROR	MEM OVERFLOW	Edit file overflowed available memory
	CS ERR	Illegal edit command
	PARAM ERR	Edit command parameter illegal
	SEQ ERR	Parameter ≤ a previous parameter or greater than the range of the symbolic file
	/I ERR	No source statements inserted after /I
	/R ERR	No replacement statements input after /R
	/C OVF	Character overflow (i.e., >72) in edit statement
	DISK OVF	No disc space for file
	FILE UN	Undefined symbolic file
RTE RELOCATING LOADER ERROR MESSAGES (Continued)	LØ1	Checksum error
	LØ2	Illegal record
	LØ3	Memory overflow

Table 4-14. Summary of RTE Error Messages (Continued)

Source	Message	Meaning
RTE RELOCATING LOADER ERROR MESSAGES (Continued)	LØ4	Base page linkage area overflow
	LØ5	Symbol table area overflow
	LØ6	Common block error a. Replacement/addition exceeds allocation b. First program in normal background load didn't declare largest common block
	LØ7	Duplicate entry points
	LØ8	No transfer address
	LØ9	Record out of sequence
	L1Ø	Operator request parameter error
	NO BLANK ID SEGMENTS	No available ID Segment
	WAITING FOR DISC SPACE	Track allocation cannot be made
	UNDEFINED EXTS	Undefined externals
	LOAD	End-of-tape condition detected
	DUPLICATE PROG NAME	Loader changes name of current program by replacing the first two characters with "##"

FMP ERROR MESSAGES

FMP errors, as shown in Table 4-15, may cause FMGR to abort (return control to the RTE system) and leave files that are empty or otherwise incomplete on the system. To purge these files, an ON,FMGR is issued to regain control, and the PUrge request is input.

Except in the above case, an error causes a transfer to the log device without aborting FMGR. Once the transfer is made to the log device, inputs are recognized only from the log device. As many requests as desired may be entered from the log device followed by a TR request (no parameters) to return to the statement following the erroneous statement.

Table 4-15. Summary of FMP Error Messages

Source	Error Number	Meaning
FMGR	ØØ1	Disc logical unit error
	ØØ2	Initialize LU2
	ØØ3	Initialize LU2
	ØØ4	Illegal response to ØØ2 or ØØ3
	ØØ5	Required track not available
	ØØ6	FMGR is suspending itself
	ØØ7	Checksum error
	ØØ8	D.RTR not found in ID segments
	ØØ9	ID segment not found (no implied transfer)
	Ø1Ø	Input error, re-enter statement caused by: Missing initial colon (non-TTY input) Supplied initial colon (TTY input) Command undefined ASCII subparameter in subparameters 3 through 5 Subparameters in other than first two parameters More than 5 subparameters Command too long
	Ø11	Some system ID segments point to the disc to be packed
	Ø12	Duplicate logical unit or label
	Ø13	Transfer stack overflow
(Continued)	Ø14	Program not found in system ID segments. Also, RP, no blank ID segments.

Table 4-15. Summary of FMP Error Messages (Continued)

Source	Error Number	Meaning
FMGR (Continued)	Ø15	Logical Source track report follows (NO implied transfer).
	Ø16	The named file is not a logical unit 2 or 3.
	Ø17	The named ID segment was not set up by FMGR.
	Ø18	The named ID segment shows the program is not dormant.
	Ø19	Checksum or setup code failed. File was not set up by an SProgram command on the current system.
	ØØØ	The given logical unit is illegal (creating a type Ø file).
	Ø21	One of the specified discs is not mounted or both specifications refer to the same disc.
	Ø22	The copy has been terminated. (NOTE: This is true of most copy errors regardless of this message being printed.)
	Ø23	The given program name is already defined in a system ID segment.
	Ø24 - Ø49	Not defined
	Ø5Ø	Illegal number of parameters
	Ø51	Illegal master security code
	Ø52	Wrong logical unit. In response to ØØ2 or ØØ3, or no disc on given LU.
	Ø53	Illegal disc label. Reference label must be positive non-zero integer. Information label must be a legal file name.
	Ø54	Disc not mounted
	Ø55	Missing parameter
	Ø56	Bad parameter. No file tracks (i.e., all directory or last track below first track). ASCII code undefined.
(Continued)	Ø57	Bad track error. Track not within file area or track is in directory area.

Table 4-15. Summary of FMP Error Messages (Continued)

Source	Error Number	Meaning
FMGR (Continued)	Ø58	Load-and-go area is undefined or empty.
	Ø59	Track not available for re-INitialization (aborts Initialization).
	Ø6Ø	INitialization will cause loss of all files on this disc. Do you really want to? Answer: YES or NO. Caused by: First track is larger. Directory will extend into a file.
FMP	≥Ø	No error detected
	- 1	Disc down. Possible interface routine cause: CREAT, PURGE, OPEN, CLOSE, READ, WRITE, APOSITION, REWIND, POSITION, RENAME
	- 2	Duplicate name. Interface routines: CREAT, RENAME
	- 3	Backspace not legal (type Ø). Interface routine: POSITION
	- 4	File too long or REC size error (type 2). Interface routine: CREAT
	- 5	Attempt to read or position to a record not written, or on update. Write an illegal record length. Interface routines: READ, WRITE, APOSITION, POSITION
	- 6	Cartridge not found or file not found or no room. Interface routines: CREAT, PURGE, OPEN, CLOSE, WRITE
	- 7	Invalid security code. Interface routines: PURGE, OPEN, WRITE, RENAME
	- 8	File currently open: eight PGM or exclusive or lock rejected. Interface routines: PURGE, OPEN, WRITE, RENAME
	- 9	Attempt to open type Ø as type 1 or to use APOSN on type Ø. Interface routines: OPEN, APOSITION
	-10	Not enough parameters. Interface routines: CREAT, PURGE, OPEN, CLOSE, READ, WRITE, LOCATE, APOSITION, POSITION, RENAME
(Continued)	-11	DCB not open. Interface routines: CLOSE, READ, WRITE, LOCATE, APOSITION, REWIND, POSITION, CENTRAL

Table 4-15. Summary of FMP Error Messages (Continued)

Source	Error Number	Meaning
FMP (Continued)	-12	SOF or EOF read or sensed. Interface routines: READ, WRITE, APOSITION, POSITION, CONTROL
	-13	Cartridge locked. Interface routines: CREAT, PURGE, OPEN, RENAME
	-14	Directory full. Interface routines: CREAT, WRITE
	-15	Illegal name. Interface routines: CREAT, RENAME
	-16	Illegal type or size = Ø (CREAT). Interface routines: CREAT, PURGE
	-17	Attempt to read or write on type Ø which does not support the operation. Interface routines: WRITE, POSITION
	-101	Illegal parameter in D.RTR Call. Possible operator error.
	-102	Illegal D.RTR call sequence (lock not requested first or file not opened exclusively first). Possible operator error.

CENTRAL INITIALIZATION

1. Loading the central system assumes that RTGEN is complete and all programs are loaded on the central disc.
2. Main power on, all peripherals "ON" or "ON-LINE."
3. Mount the system disc and set the switch to "LOAD."
4. Load the RTE Boot Tape into the Paper Tape Reader (to boot RTE into core from the disc using BBL):
   a. Press SWITCH REGISTER.
   b. Clear DISPLAY REGISTER.
   c. Press P-REGISTER.
   d. Set DISPLAY REGISTER to $n7700_8$.

      where n = 5 - 24K
      7 - 32K

   e. Press EXTERNAL PRESET, INTERNAL PRESET, LOADER ENABLE and RUN.
5. Press SWITCH REGISTER and clear the DISPLAY REGISTER.
6. Press P-REGISTER and clear the DISPLAY REGISTER.
7. Set the DISPLAY REGISTER 100_8.
8. Press EXTERNAL PRESET, INTERNAL PRESET and RUN.
9. Turn on the 91701 package.
   a. Power "ON" terminal computers.
   b. Enable the central computer system console.
   c. Type in at the central system console:

      ON,LSTEN *p1, p2, p3* (CR) (LF)
10. See Section VI for LSTEN description.

SECTION V

PROGRAM-TO-PROGRAM COMMUNICATION

Application programs running at a terminal computer may communicate directly with application programs running at the central computer via program-to-program communication. The vehicles for this dialogue are the D.65 BCS Communications driver at the terminal and the DVR65 RTE Communications driver at the central computer. Figure 5-1, below, is a block diagram of the program-to-program communication process. The arrows in the figure represent transmissions across the data link provided by the communications drivers.

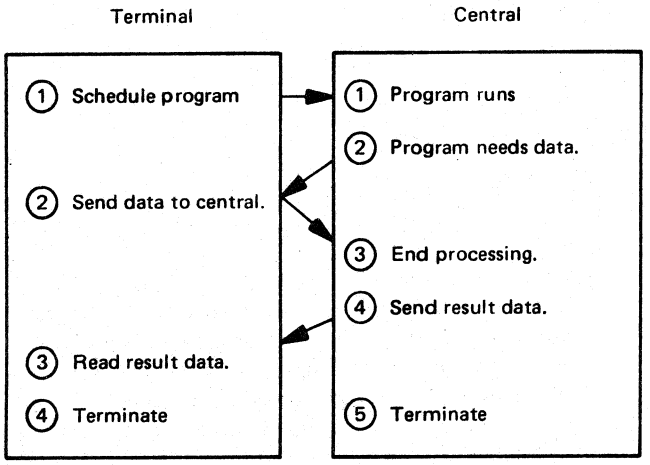


Figure 5-1. Block Diagram of Program-to-Program Communication

The central computer program is remotely scheduled by the terminal and obtains the processing parameters from the terminal, performs the processing, and sends the request to read (write) data back to the terminal. The calling sequences required by the D.65 and DVR65 drivers to communicate across the data link between computers are described in Appendix A.

The process in Figure 5-1 is expanded in Figure 5-2 to show the sequence in which program calls are made.

An example of the use of the program-to-program feature is the case where a small terminal computer is used in a test cell to collect large amounts of data that need extensive processing before it can be output on a central disc file or the central line printer. In this case, the data collection rate at the terminal is so fast that there is no time left to process data. The solution to the problem is for the terminal computer to collect the data and pass it to the larger, more powerful, central computer for processing.

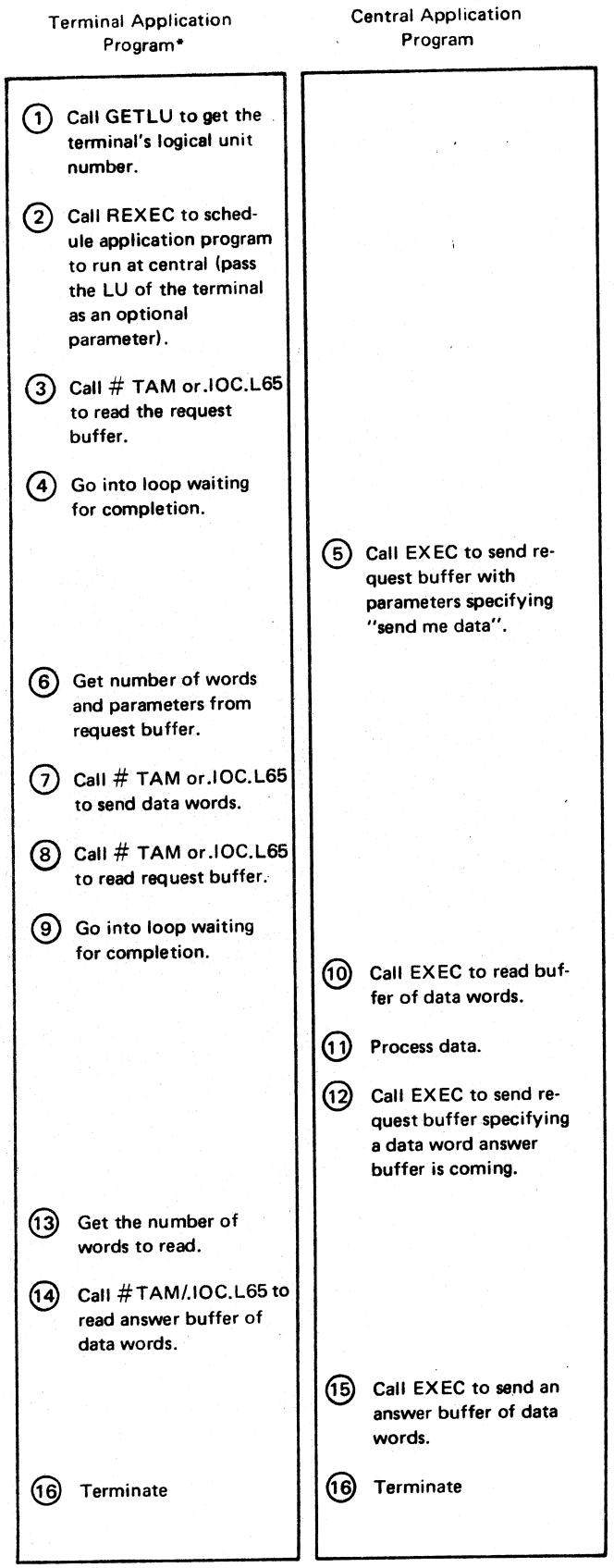


Figure 5-2. Program Calls for Program-to-Program Communication

* #TAM and .IOC. are not callable from FORTRAN.

The material that follows describes the programming considerations required to implement the program-to-program feature for use with terminal data collection and central processing. The discussion is concerned primarily with terminal programs running under TCE/3. Some exceptions are noted for TCE/1 and TCE/2 usage.

If the terminal program is designed to run under TCE/3, .IOC. and D.65 are core resident within the TCE/3 executive and are callable by application programs. If the terminal program is to be run under either TCE/1 or TCE/2, it must be fully stand-alone, with its own I/O environment including .IOC. and D.65. The SIO and BCS System manuals should be consulted along with the "System Generation" section of this manual.

In order to achieve the degree of interaction required between the terminal program and the central program, they should be designed in parallel, ideally by the same person. The central computer functions as a slave to the terminal computer's processing requests – which means that the terminal computer must initiate all processing.

The Distributed System terminal computer initiates the dialogue by causing the central computer to schedule its dialogue program. If TCE/1 or TCE/2 are in the terminal, this must be done manually by the central operator. If TCE/3 is in the terminal, ESCAPE function processing should be disabled, and the logical unit number of the terminal obtained before a Remote EXEC call is issued to schedule the dialogue program at the central computer. Steps 1 and 2 of Figure 5-2 show the program calls made under TCE/3 to initiate dialogue. The "Terminal Receives Data" and "Terminal Sends Data" program samples at the end of this section show how these steps can be coded.

As mentioned above, the first task of the terminal program running under TCE/3 is to disable processing of the Escape function. **Pressing the ESC (Escape) key during Program-to-Program Communication can cause a loss of all data communication paths to central (if DVR65 at central is transmitting to the terminal).** The Escape function can be disabled by either calling the utility subroutine, PTPON, or by setting the $BUSY flag to a non-zero value directly. The sample programs show the code for the call to PTPON. Manually setting the $BUSY flag is discussed in Section IV. A call to PTPOF to clear the $BUSY flag and reenable Escape processing should be made before Step 4 of Figure 5-1 is reached.

Terminal processing of a previously pressed ESC key may be done at "safe" points in the program-to-program communication sequence. After a terminal program has read a request and before it receives or transmits data, a program call may be made to ESCON. If the ESC key has been pressed, the terminal program transmits a STOP reply to central which causes the data transfer from central to be refused and the central program to terminate. The terminal program then terminates itself.

The user program at central must be programmed in such a way that upon "completion" of the expected data transfer, a check is made on bit 3 of the EQT status word contained in the A-Register (see Appendix A). If the bit is set, the central program should terminate itself.

Once Escape processing is disabled, the terminal program issues a program call to the utility subroutine GETLU (shown in Step 1 of Figure 5-2). This call is made because the program to be scheduled at the central computer needs to know the logical unit of the terminal with which it will communicate. The logical unit (obtained from the call to GETLU) is passed to the central program as one of the optional parameters in the remote schedule call shown in the program samples. This parameter need not be passed, if the logical unit is fixed at a known value and hard-coded into the central program.

Once the logical unit number is obtained, the terminal program can make the call to REXEC (Step 2 of Figure 5-2) for a remote schedule request. Additional parameters may be passed to the central program at this time. After the call is completed, the terminal program may perform the Remote Request Error Checks shown in Figure 5-3. The remote communication status is obtained by checking the ISTAT parameter. Whether the central program actually got scheduled is determined by checking the A-Register (see the REXEC Program schedule call description in Section IV). If the central program was not dormant (already scheduled or suspended), the schedule call should be repeated until the central program becomes available.

Even though the central program is scheduled to run, there is no guarantee that the program will run immediately. It may have a lower priority than other scheduled programs and have to wait its turn to run. The only way, therefore, that the terminal program can know when the central program is actually ready to communicate is for the terminal program to make a call to receive request and the central program to make a call to send request (Steps 3, 4 and 5 of Figure 5-2). I/O completion does not occur until the central program finally begins execution and sends the first request buffer.

A direct program-to-program communication data link is established between the terminal program and the central program when the first buffer is sent. All ensuing conversation between programs takes place via buffers as shown in Steps 6 - 16 of Figure 5-2 and in the remainder of the code in the sample programs. The contents of the buffers, i.e., conversation contents, are determined by the user. The material which follows describes some of the programming considerations resulting from the format descriptions of DVR65 and D.65 detailed in Appendix A. The use of #TAM calls to replace D.65 calls is also described.

Transmission between the central computer and the terminal takes place in either of two submodes: transmit/receive

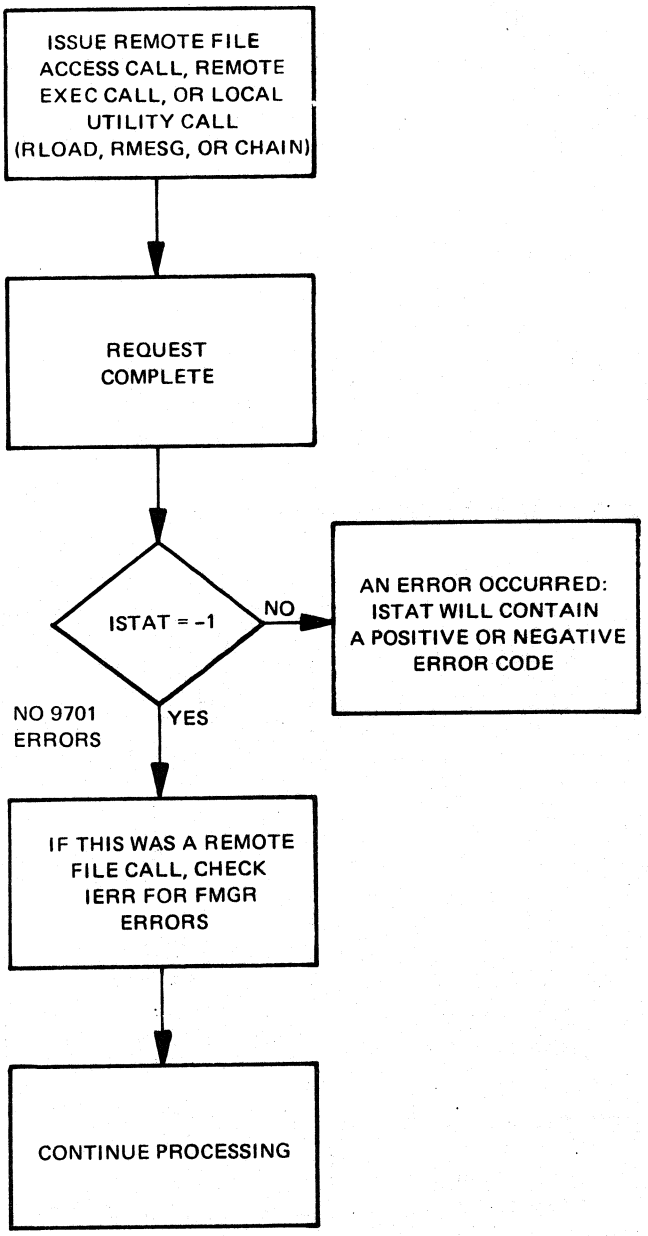


Figure 5-3. Remote Request Error Check

"request," or transmit/receive "data." Transmit/receive "data" is the secondary submode. The content of the request buffer is determined by the user. The buffer can be any length (including zero), but it normally contains user program-to-program parameters. Data transmissions are normally confined to the transmit/receive "data" submode.

A "transmit request" call is made by the user program at central, and **NEVER** from the terminal program. This is due to the fact that requests sent to central do not go to the central application program, but to the resident QUEUE module which expects the specific format used for remote file access parameter buffers. If a "transmit request" call is not made in the required format, the results are disastrous. Conversely, a "receive request" can be made only by a terminal program.

A "transmit data" call can be made by either the terminal program or by the central program, as long as the central program initiates the data transmission with a "transmit request" call. The computer that receives the data then performs a "receive data" call. For every data transfer, regardless of direction, central must first send a "request" to the terminal (see Figure 5-2 and sample programs).

Request and data buffer lengths must be exactly the same at both the terminal and at central. If the number of words read is smaller than the number of words sent, the communications drivers will loop trying to send the rest of the buffer. Request buffer lengths are predictable and fixed by the user to known values. Data buffer lengths are usually variable, so that the length of the buffer to be transferred is normally conveyed by the central program in an optional parameter in the request buffer. Data buffer lengths cannot be zero. If the length is sent as a parameter, it is used to configure the "transmit/receive data" call.

The "receive request" call at the terminal may be made in either Listen mode, or Polling mode. These modes should be established in the user's program before the "receive request" call is made (see enable Listen mode call and enable Polling mode call in Section IV). If Listen mode is used, when the central program transmits a request, a user-written, BCS subroutine is executed at the terminal upon the receipt of the interrupt from central. This subroutine must have an entry point called INT65. See Appendix A for further details.

If Polling mode is used, the terminal user program checks the communication card with a "read request" call. If no request is received, bit 6 of the EQT status word is set (see Appendix C), and the "read request" call must be re-issued. If a request has been received, bit 6 is zero, and processing may continue. See Appendix A for further details.

The TCE/3 resident module #TAM may be used instead of .IOC. calls for terminal program-to-program communications written in HP Assembly Language. The calls to #TAM are

described in Section IV. #TAM converts user calls to the appropriate .IOC. calling sequence for D.65. It automatically determines and sets up the proper transmission mode (interrupt open loop or DMA open loop), and provides polling mode for "receive request" calls.

IMPORTANT

1. A central user program can only **transmit a request.** A terminal program can only **receive a request.**

2. Request or data buffer length at the terminal must be exactly the same as the corresponding request or data buffer length at central.

3. Protection from ESC key processing should be provided by a call to PTPON before program-to-program communication begins.

SAMPLE PROGRAM

```
<<<<<<<<<<<<<<<<<<<<<<<<<< TERMINAL RECEIVES DATA >>>>>>>>>>>>>>>>>>
******* TERMINAL PROGRAM ******* *  ***** CENTRAL PROGRAM ABCD *****
                                 *
     EXT PTPON,PTPOF,ESCON,IDLE  *
     EXT GETLU,REXEC,#TAM        *
     .                           *
     .                           *
     .                           *
     JSB PTPON  DISABLE OPERATOR *
     DEF *+1    ATTENTION.       *
                                 *
     JSB GETLU  GET LOGICAL UNIT *
     DEF *+2    AT CENTRAL FOR   *
     DEF LU     THIS TERMINAL.   *
                                 *
SCHD JSB REXEC  SCHEDULE ABCD    *
     DEF *+5    AT CENTRAL.      *
     DEF STATS                   *
     DEF B12                     *
     DEF PGNAM  ASCII "ABCD  ".  *       .
     DEF LU     PASSED PARAMETER.*       .
                                 *       .
     SZA        ABCD DORMANT?    *       LDA 1,I    GET PASSED LU.
     JMP SCHD   NO, RE-SCHEDULE. *       STA LU
                                 *
     CLA,INA    READ REQUEST     *       JSB EXEC   SEND REQUEST
     JSB #TAM   FROM CENTRAL.    *       DEF *+6    AND DATA.
     DEF *+5                     *       DEF B2     REQ. CODE = WRITE.
     DEC 5                       *       DEF LU     LU OF TERMINAL.
     DEF BUFFR                   *       DEF PBUFR  PARAM BUFFER.
     DEC -2     1 WORD REQUEST.  *       DEF B4     PARAM BUFFER LENGTH.
     DEF *+1    COMPLETOR SUBR.  *       DEF B10    TRANSMISSION MODE.
                                 *
     NOP                         *       SLA,RSS    ANY ERRORS?
     JSB ESCON  CHECK IF OPERATOR*       JMP ERROR  YES.
     DEF *+2    HAS PRESSED      *       .          NO, CONTINUE
     DEF TEMP   ESCAPE KEY.      *       .          PROCESSING.
     LDA TEMP                    *       .
     SZA                         *ERROR  ALF,ALF    WAS STOP COMMAND
     JMP XSTP   YES, SEND STOP.  *       ALF,SLA    RECEIVED?
                                 *       JMP DONE   YES, TERMINATE.
     CLA        READ DATA FROM   *       .          NO, CHECK FOR
     JSB #TAM   CENTRAL.         *       .          OTHER ERRORS.
     DEF *+5                     *       .
     OCT 5                       *
     DEF BUFFR                   *
     DEC -40    20 WORD RECORD.  *
     DEF *+1    COMPLETOR SUBR.  *
                                 *
     .          READ & PROCESS   *       .
     .          TILL DONE.       *       .
                                 *       .
DONE JSB PTPOF  ENABLE OPERATOR  *DONE   JSB EXEC   TERMINATE ABCD.
     DEF *+1    ATTENTION.       *       DEF *+2
     JSB IDLE   RETURN TO TCE/3. *       DEF B6
                                 *
XSTP JSB #TAM   TRANSMIT STOP    *PBUFR  DEF DBUFR  DATA BUFFER.
     DEF *+2      TO ABCD.       *       DEC 20      (20 WORDS)
     DEC 2                       *       DEF RBUFR  REQUEST BUFFER.
     JMP DONE   EXIT.            *       DEC 1       (1 WORD)
```

SAMPLE PROGRAM

```
<<<<<<<<<<<<<<<<<<<<<<<<<< TERMINAL SENDS DATA >>>>>>>>>>>>>>>>>>>>>
******* TERMINAL PROGRAM ******* *  ***** CENTRAL PROGRAM ABCD *****
                                 *
     EXT PTPON,PTPOF,ESCON,IDLE  *
     EXT GETLU,REXEC,#TAM        *
     .                           *
     .                           *
     .                           *
     JSB PTPON  DISABLE OPERATOR *
     DEF *+1    ATTENTION.       *
                                 *
     JSB GETLU  GET LOGICAL UNIT *
     DEF *+2    AT CENTRAL FOR   *
     DEF LU     THIS TERMINAL.   *
                                 *
SCHD JSB REXEC  SCHEDULE ABCD    *
     DEF *+5    AT CENTRAL.      *
     DEF STATS                   *
     DEF B12                     *
     DEF PGNAM  ASCII "ABCD  .   *       .
     DEF LU     PASSED PARAMETER.*       .
                                 *       .
     SZA        ABCD DORMANT?    *       LDA 1,I    GET PASSED LU.
     JMP SCHD   NO, RE-SCHEDULE. *       STA LU
                                 *
     CLA,INA    READ REQUEST     *       JSB EXEC   SEND REQUEST
     JSB #TAM   FROM CENTRAL.    *       DEF *+6    AND RECEIVE DATA.
     DEF *+5                     *       DEF B1     REQ. CODE = READ.
     DEC 5                       *       DEF LU     LU OF TERMINAL.
     DEF BUFFR                   *       DEF PBUFR  PARAM BUFFER.
     DEC -2     1 WORD REQUEST.  *       DEF B4     PARAM BUFFER LENGTH.
     DEF *+1    COMPLETOR SUBR.  *       DEF B0     TRANSMISSION MODE.
                                 *
     NOP                         *       SLA,RSS    ANY ERRORS?
     JSB ESCON  CHECK IF OPERATOR*       JMP ERROR  YES.
     DEF *+2    HAS PRESSED      *       .          NO, CONTINUE
     DEF TEMP   ESCAPE KEY.      *       .          PROCESSING.
     LDA TEMP                    *       .
     SZA                         *ERROR  ALF,ALF    WAS STOP COMMAND
     JMP XSTP   YES, SEND STOP.  *       ALF,SLA    RECEIVED?
                                 *       JMP DONE   YES, TERMINATE.
     CLA        SEND DATA TO     *       .          NO, CHECK FOR
     JSB #TAM   CENTRAL.         *       .          OTHER ERRORS.
     DEF *+4                     *       .
     OCT 6                       *
     DEF BUFFR                   *
     DEC -40    20 WORD RECORD.  *
                                 *
                                 *
     .          READ & PROCESS   *       .
     .          TILL DONE.       *       .
                                 *       .
DONE JSB PTPOF  ENABLE OPERATOR  *DONE   JSB EXEC   TERMINATE ABCD.
     DEF *+1    ATTENTION.       *       DEF *+2
     JSB IDLE   RETURN TO TCE/3. *       DEF B6
                                 *
XSTP JSB #TAM   TRANSMIT STOP    *PBUFR  DEF DBUFR  DATA BUFFER.
     DEF *+2      TO ABCD.       *       DEC 20      (20 WORDS)
     DEC 2                       *       DEF RBUFR  REQUEST BUFFER.
     JMP DONE   EXIT.            *       DEC 1       (1 WORD)
```

SECTION VI

SYSTEM GENERATION AND INSTALLATION

This section is divided into four parts that describe the system generation and installation procedures for the Distributed System.

PART I
TCE/1 GENERATION AND INSTALLATION

The generation of TCE/1 involves loading an absolute assembly language program directly into the computer either by front panel switches or reading paper tapes.

PART II
TCE/2 GENERATION AND INSTALLATION

TCE/2 is an absolute assembly language program. It is available in separate versions for 4K, 8K, 16K, 24K and 32K terminal computers. TCE/2 is loaded into the terminal computer from the central disc, or from paper tape via BBL.

PART III
TCE/3 GENERATION AND INSTALLATION

TCE/3 is a collection of relocatable assembly language modules. The generation of the resident system is performed by the RTE System Cross Loader at the central computer. The relocatable modules that make up TCE/3 are TEXEC, RFAIN, #TAM, .IOC., D.ØØ, D.65 plus any other BCS peripheral or instrument drivers required by a particular configuration.

PART IV
CENTRAL GENERATION AND INSTALLATION

Generation of the 91701 software for the central computer consists of performing an RTE generation which includes the 91701 central program modules, the File Management Package and the RTE System Cross Loader, in addition to the RTE program modules. The minimum and maximum configuration examples show the central generation output for a 24K and a 32K computer.

PART I

TCE/1 GENERATION AND INSTALLATION

TCE/1 is an absolute Assembly Language program. It is loaded into terminal computers with memories larger than 4K via the BBL as described below. Computers with 4K memories require that TCE/1 be loaded via the front panel switches. The octal listing of TCE/1 in Appendix E is provided for this purpose.

1. HALT CPU.
2. Load 4K TCE/1 exchange program tape in photo reader.
3. Set P-Register to starting address of protected bootstrap area (i.e., $\emptyset m77\emptyset\emptyset$).
4. Clear SWITCH REGISTER, press PRESETS, LOADER ENABLE, and RUN.
5. If HALT 77, set P-Register again to starting address of 4K TCE/1 (07700) ~~of the bootstrap area~~. If not HALT 77 return to Step 1. ~~Replace I/O assignments as required.~~ See TCE/1 listing in Appendix E.
   Clear Location 07727
   ~~Replace I/O assignments as required~~
6. Set switch to *nn* (PROG *nn*) for loading the appropriate TCE/1 bootstrap into the protected area.
   PROGnn must have previously been stored on the central disk via the FNARcommand :ST,5,PROGnn,BA
7. Press PRESETS, LOADER ENABLE, and RUN.
8. If HALT 77, TCE/1 is now the system's bootstrap loader. If not HALT 77, return to Step 4. Replace I/O assignment code as required. See TCE/1 listing in Appendix E.

where *nn* = $\emptyset - 77_8$
m = 1 for 8K, 2 for 12K, 3 for 16K, 5 for 24K, 7 for 32K

A return from TCE/1 to the BBL is made following the instructions below.

1. HALT CPU.
2. Set P-Register to starting address of protected bootstrap area (i.e., $\emptyset m77\emptyset\emptyset$). Must be greater than 4K.
3. Set SWITCH REGISTER to *nn* (PROG *nn* for loading the 4K TCE/1 exchange program.

4. Press EXTERNAL PRESET, INTERNAL PRESET, LOADER ENABLE, and RUN, sequentially.

5. If HALT 77, set P-Register to the starting address of the 4K TCE/1 exchange program (ØØ77ØØ). If not a HALT 77, return to Step 2.

6. Set SWITCH REGISTER to *nn* (PROG *nn*) for loading the BBL.

7. Press PRESETS, LOADER ENABLE, and RUN.

8. If HALT 77, BBL is now in protected area (Ø*m*77ØØ), if not HALT 77, return to Step 5.

9. Reset I/O assignments as required.

where *nn* = Ø - 77_8
m = 1 for 8K, 2 for 12K, 3 for 16K, 5 for 24K, 7 for 32K

To change the I/O assignments originally assembled into TCE/1, the procedure below should be followed:

1. Press LOADER ENABLE (light ON).

2. For each of the following addressed, press M, set DISPLAY REGISTER to the address, press MEMORY DATA, and set bits Ø – 5 of DISPLAY REGISTER to the I/O channel of the remote computer.

*n*77ØØ	*n*7745
*n*77Ø1	*n*7754
*n*77Ø2	*n*7756
*n*7742	*n*7757
*n*7743	*n*7763

where *n* = Ø for 4K, 1 for 8K, 2 for 12K, 3 for 16K, 5 for 24K, 7 for 32K

3. Press LOADER ENABLE (light OFF).

Clear Location 7727_8
Replace I/O Assignments as required.

PART II

TCE/2 GENERATION AND INSTALLATION

TCE/2 is an absolute assembly language program. It is available in 4K, 8K, 12K, 16K, 24K and 32K versions. TCE/2 resides on the central disc as an absolute file. When it is loaded via TCE/1 or TCE/3 into the terminal computer, it occupies the upper portion of memory next to the protected area (as shown in Table B-1 of Appendix B).

TCE/2 is stored on the central disc by the RTE File Manager following the directions below.

To store TCE/2 on the central disc, place the TCE/2 absolute paper tape in the central reader and use the following commands at the central computer keyboard device:

```
* ON, FMGR
: ST, n, PROGnn, BA
```

Binary absolute record format

File name

Logical unit number of the input device (usually 5)

```
: TR        Exit from the File Manager
```

When TCE/2 is loaded into terminal memory, the assembled I/O channel assignments for the Serial Data Interface (SDI) channel and the TTY device may be changed by starting execution at P = $10\emptyset\emptyset_8$ with the S-Register bits $\emptyset$ - 5 equal to the SDI channel, and bits 6 - 11 equal to the TTY channel as shown in Figure 6-1.

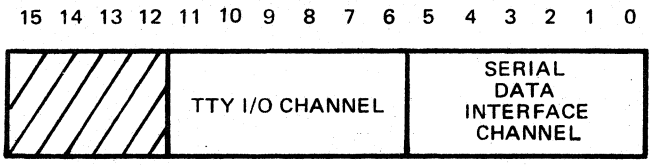


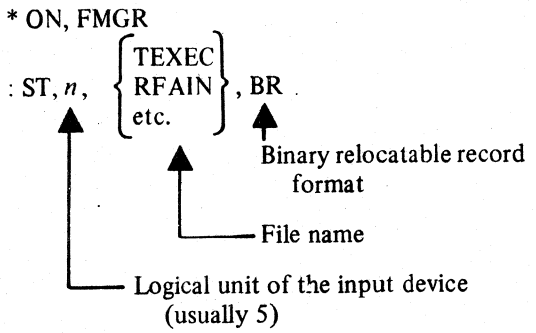
Figure 6-1. S-Register for I/O Channel Assignments

Additional information is provided in Section IV, Part I.

PART III

TCE/3 GENERATION AND INSTALLATION

TCE/3 consists of the following relocatable assembly language modules: TEXEC, RFAIN, #TAM, .IOC., D.ØØD, D.65 and any other required BCS peripheral or instrument drivers. These modules are stored on the central disc by the RTE File Manager using the following commands:



TCE/3 is generated by the RTE system cross loader either from the files stored on the central disc, or from paper tapes. Although TCE/3 is a BCS environment, SXL is used for generation instead of PCS because of residency requirements and user program interface. TCE/3 modules can be input to the loader as separate relocatable disc files, or as a combined relocatable "library" (see **RTE System Cross Loader Manual** for the appropriate commands).

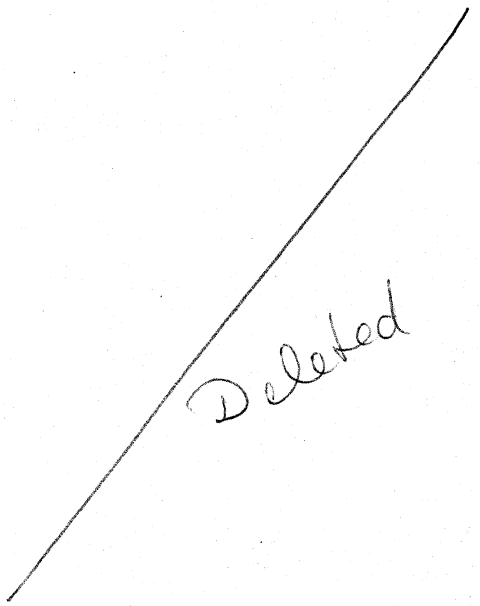
The SXL Loader command file used to generate TCE/3 contains commands to build the SQT, EQT, interrupt linkages, .MEM. table; to define memory bounds and to produce a resultant command file. This command file is saved on the central disc and used as part of the command file required to configure terminal application programs. It contains information to direct the Loader to set application program core bounds so that TCE/3 is not overlayed. It also indicates where the TCE/3 entry points are located. The Loader output must be in absolute binary format (not core image).

If the command file is on the standard input device, the Loader is turned on with an automatic transfer.

```
ON,SXL
```

If the command file is on disc, the Loader is turned on with the commands:

```
ON,SXL,1
– TR,file name
```



The example which follows shows an SXL Loader command file for a minimal TCE/3 generation. It can be modified for a particular terminal configuration. This example assumes 8K memory, TTY (or CRT) in channel 15, remote central computer in channel 13, and the use of the BCS – TTY and communications drivers (D.ØØD and D.65).

The last command in the input file is the SNAP statement. It produces the partial command file used to prepare the application programs for the terminals.

Statements preceded by a dash (–) indicate input from the system console.

```
****** T=00000 IS ON LU 09

0001  -ECHO ON 6
0002  * DEFINE MEMORY BOUNDS FOR TCE/3
0003  -BOUNDS FWAM=2000,LWAM=17677,FWABP=105,LWABP=1777
0004  * PLACE OUTPUT ON FILE ACCESSABLE TO TCE/2 AND TCE/1
0005  -OUTPUT ABS ON PROG10
0006  -MAP ALL
0007  -RELOCATE TEXEC,RPAIN,#TAM,.IOC.,D.00D,D.65
0008  * (THE ABOVE FILES COULD ALL HAVE BEEN COMBINED
0009  *    IN A SINGLE "LIBRARY" FILE SUCH AS "TCELIB")
0010  * BUILD BCS SQT (STANDARD EQUIPMENT TABLE)
0011  -SET .SQT. TO LOCC
0012  -SET $(LOCC) TO 7
0013  -SET $(LOCC+1) TO 7
0014  -SET $(LOCC+2) TO 7
0015  -SET $(LOCC+3) TO 7
0016  -SET $(LOCC+4) TO 7
0017  -SET $(LOCC+5) TO 7
0018  * DEFINE # OF EQT (EQUIPMENT TABLE) ENTRIES
0019  -SET .EQT. TO LOCC+6
0020  -SET $(LOCC+6) TO 2
0021  * BUILD LU 7 EQT FOR D.00 (TTY ON CHANNEL 15B)
0022  -SET $(LOCC+7) TO 15
0023  -SET $(LOCC+10) TO 0
0024  -SET $(LOCC+11) TO 6
0025  -SET $(LOCC+12) TO D.00
0026  * BUILD LU 10B EQT FOR D.65 (CHANNEL 13B)
0027  -SET $(LOCC+13) TO 13
0028  -SET $(LOCC+14) TO 32400
0029  -SET $(LOCC+15) TO 0
0030  -SET $(LOCC+16) TO D.65
0031  * SET UP I/O INTERRUPT LINKAGES
0032  -SET $(15) TO 114025
0033  -SET $(25) TO I.00
0034  -SET $(13) TO 114023
0035  -SET $(23) TO I.65
0036  * DEFINE MEMORY AVAILABLE TO A USER PROGRAM
0037  -SET FWAM TO LOCC+30
0038  -SET FWABP TO BPLOCC
0039  -SET .MEM. TO LOCC+17
0040  -SET $(LOCC+17) TO LOCC+20
0041  -SET $(LOCC+20) TO FWABP
0042  -SET $(LOCC+21) TO LWABP
0043  -SET $(LOCC+22) TO FWAM
0044  -SET $(LOCC+23) TO LWAM
0045  * SET UP SQT, EQT POINTERS
0046  -SET $(XSQT) TO .SQT.
0047  -SET $(XEQT) TO .EQT.
```

Deleted

```
0048   * ADJUST LOCC TO FWA USER MEMORY
0049   -SET LOCC TO FWAM
0050   -DISPLAY UNDEFS ON 6
0051   -DISPLAY UNDEFS
0052   * SAVE SNAP FOR USE IN TERMINAL USER PROGRAM PREPARATION
0053   -SNAP ON 4
0054   -END
```

The SXL output from the TCE/3 generation is shown below. The information output includes core locations of the generated modules and entry point addresses.

```
****** T=00000 IS ON LU 09

0001    -* DEFINE MEMORY BOUNDS FOR TCE/3
0002    -=BOUNDS FWAM=2000,LWAM=17677,FWABP=105,LWABP=1777
0003    -* PLACE OUTPUT ON FILE ACCESSABLE TO TCE/2 AND TCE/1
0004    -=OUTPUT ABS ON PROG10
0005    -=MAP ALL
0006     FILE   PROGRAM GLOBAL   FIRST    LAST       BASE PAGE     REF.BY
0007     NAME   MODULE  VARS.    ADDRS.   ADDRS.
0008     ------------------------------------------------------------------
0009    -=RELOCATE TEXEC,RFAIN,=TAM,IOC,0.00D,0.65
0010     TEXEC
0011           TEXEC             002000   004562   000105   000321
0012                   ATTEN     002102
0013                   RLOAD     003223
0014                   CHAIN     003611
0015                   RNPGM     003515
0016                   RMESG     003527
0017                   IDLE      002000
0018                   PTPON     003677
0019                   PTPOF     003704
0020                   ESCON     003711
0021    RFAIN
0022           RFAIN             004563   006444   000322   000432
0023                   .ENTR     005437                      TEXEC
0024                   $BUSY     006037                      TEXEC
0025                   RFATM     004563                      TEXEC
0026                   GETLU     004600                      TEXEC
0027                   RCRET     004647                      TEXEC
0028                   RPURG     004676                      TEXEC
0029                   ROPEN     004710
0030                   RREAD     004723
0031                   RWRIT     004742
0032                   RPOSN     005013
0033                   RWIND     005025
0034                   RCLOS     005035                      TEXEC
0035                   RNAME     005046                      TEXEC
0036                   RCONT     005062
0037                   RLOCF     005074
0038                   RAPOS     005104
0039                   RSTAT     005117
```

Deleted

```
0040                    REXEC   005137                                  TEXEC
0041    #TAM
0042            #TAM            006445  007077  000433  000432
0043                    #TAM    006445                                  TEXEC
0044                    INT05   007025
0045    IOC
0046            IOC             007100  007316  000433  000432
0047                    .IOC.   007100                                  TEXEC
0048                    DMAC1   007315
0049                    DMAC2   007316
0050                    IOERR   007274
0051                    XSQT    007313
0052                    XEQT    007314                                  TEXEC
0053                    .BUFR   007246
0054    D.00D
0055            D.00D           007317  010003  000433  000436
0056                    D.00    007317
0057                    I.00    007451
0058                    $ESC    007742                                  TEXEC
0059    D.65
0060            D.65            010004  012117  000437  000476
0061                    D.65    010004
0062                    I.65    011003
0063    -* (THE ABOVE FILES COULD ALL HAVE BEEN COMBINED
0064    -*    IN A SINGLE "LIBRARY" FILE SUCH AS "TCELIB")
0065    -* BUILD BCS SQT (STANDARD EQUIPMENT TABLE)
0066    --SET .SQT. TO LOCC
0067                    .SQT.   012120
0068    --SET $(LOCC) TO 7
0069    --SET $(LOCC+1) TO 7
0070    --SET $(LOCC+2) TO 7
0071    --SET $(LOCC+3) TO 7
0072    --SET $(LOCC+4) TO 7
0073    --SET $(LOCC+5) TO 7
0074    -* DEFINE # OF EQT (EQUIPMENT TABLE) ENTRIES
0075    --SET .EQT. TO LOCC+6
0076                    .EQT.   012126                                  TEXEC
0077    --SET $(LOCC+6) TO 2
0078    -* BUILD LU 7 EQT FOR D.00 (TTY ON CHANNEL 15B)
0079    --SET $(LOCC+7) TO 15
0080    --SET $(LOCC+10) TO 0
0081    --SET $(LOCC+11) TO 0
0082    --SET $(LOCC+12) TO D.00
0083    -* BUILD LU 10B EQT FOR D.65 (CHANNEL 13B)
0084    --SET $(LOCC+13) TO 13
0085    --SET $(LOCC+14) TO 32400
0086    --SET $(LOCC+15) TO 0
0087    --SET $(LOCC+16) TO D.65
0088    -* SET UP I/O INTERRUPT LINKAGES
0089    --SET $(15) TO 114025
0090    --SET $(25) TO I.00
0091    --SET $(13) TO 114023
0092    --SET $(23) TO I.65
0093    -* DEFINE MEMORY AVAILABLE TO A USER PROGRAM
0094    --SET FWAM TO LOCC+30
0095                    FWAM    012150
```

Deleted

```
0096    --SET FWABP TO BPLOCC
0097                    FWABP   000477
0098    --SET .MEM. TO LOCC+17
0099                    .MEM.   012137                                  TEXEC
0100    --SET $(LOCC+17) TO LOCC+20
0101    --SET $(LOCC+20) TO FWABP
0102    --SET $(LOCC+21) TO LWABP
0103    --SET $(LOCC+22) TO FWAM
0104    --SET $(LOCC+23) TO LWAM
0105    -* SET UP SQT, EQT POINTERS
0106    --SET $(XSQT) TO .SQT.
0107    --SET $(XEQT) TO .EQT.
0108    -* ADJUST LOCC TO FWA USER MEMORY
0109    --SET LOCC TO FWAM
0110                    LOCC    012150
0111    --DISPLAY UNDEFS ON 6
0112    NO UNDEFS
0113    --DISPLAY UNDEFS
0114    -* SAVE SNAP FOR USE IN TERMINAL USER PROGRAM PREPARATION
0115    --SNAP ON 4
0116    --END
0117    NO UNDEFS
0118    LINKS TABLE
0119     ATTEN AB000332
0120     .ENTR   000321
0121     $BUSY   000120
0122     RFATM   000205
0123     GETLU   000227
0124     RCRET   000223
0125     RPURG   000226
0126     RCLOS   000225
0127     RNAME   000224
0128     REXEC   000207
0129     #TAM    000230
0130     INT65   000442
0131     .IOC.   000122
0132     DMAC1   000437
0133     DMAC2   000440
0134     IOERR   000441
0135     XEQT    000256
0136     $ESC    000121
0137     .EQT.   000255
0138     .MEM.   000105
0139    ****** END LOADING ******
0140    TRANSFER TO  TEXEC   002550
0141    CORE USED    002000  012143  000013  000476
0142    FREE CORE    012144  077677  000477  001777
0143    OUTPUT FILE NAME:PROG10
0144    ****** RELOCATION COMPLETED SUCCESSFULLY ******
```

Deleted

Locations 2 and 3 of the terminal computer contain a JMP 3,I and the starting address of the last program loaded. When TCE/3 is initially loaded, location 3 contains the TCE/3 start address. Any subsequent program load changes the contents of these locations to the address of the new program.

TCE/3 may be manually restarted at location 2000_8. This causes a transfer to the abort section of TCE/3 and has the same effect as an ABORT command used on an application program.

PART IV

CENTRAL GENERATION AND INSTALLATION

The central 91701 software is generated as part of the standard RTE Generation (RTGEN) process described in the **Real-Time Executive Software System Manual**. This process consists of four phases: (1) Initialization Phase, (2) Program Input Phase, (3) Parameter Input Phase, and (4) Disc Loading Phase. HP 91701 central software is configured into the RTE System during each of these phases. When RTGEN is complete and the system booted, the HP 91701 module LSTEN is scheduled (to complete the communication link between computers), and the terminals are initialized as described in Parts I, II, and III of this Section. Sample configuration work sheets and printouts of the RTGEN process are provided for maximum and minimum HP 9701 Systems in addition to the descriptions which follow.

INITIALIZATION PHASE

The information required by RTGEN is not modified by HP 91701. The user must provide a track map of the system disc space, memory, time base generator channel, swapping option, and program input devices as described in the RTE Manual and as shown in the sample work sheets which follow.

It should be remembered that 24K systems require 24K SIO drivers, and 32K systems require ~~32K~~ 24 SIO drivers.

PROGRAM INPUT PHASE

The relocatable programs for HP 91701 FMP and the SXL Loader (if used) are configured into the system during this phase. Programs are loaded in the following order:

HP 91701 Exec Control (EXEC)

Scheduler (SCHED)

I/O Control (RTIOC)

I/O Drivers (The 24K version of the HP 91701 central cannot accommodate all of the RTE drivers. See 24K core map in Appendix B for driver trade offs.)

System programs written by the user

Foreground core-resident programs

- QUEUE
- DISP

Foreground disc-resident programs

- TAM*
- RFAM
- DISC
- ERR*
- PROGL + ~~BCKOS~~ Deleted
- DLIST

*Foreground core-resident where space allows.

Background disc-resident programs

Assembler (main and its segments)

FORTRAN (main and its segments) and/or FORTRAN

FORTRAN IV (main and its segments, but not both FORTRAN IV versions)

ALGOL (main and segment)

RTE Relocating Loader

Editor

SXL Loader

FMGR

Other background disc-resident programs and their respective segments, if any

LSTEN

Library programs

QUDIS

Utility programs

The central computer portion of the HP 91701 software is configured into a 32K RTE System with the parameters shown in Table 6-1. The parameter overrides required for a 24K system are listed under the PARAMETER INPUT PHASE.

Table 6-1. 32K RTGEN Parameter Summary for HP 91701 Modules

Module Name	Type	Default Residency	Default Priority
D.RTR	1	Foreground core	1
QUEUE	1	Foreground core	2Ø
QUDIS	6	Library	none
DISP	1	Foreground core	10
TAM	2	Foreground disc	30
RFAM	2	Foreground disc	30
DISC	2	Foreground disc	30
ERR	2	Foreground disc	30
LSTEN	3	Background disc	80
PROGL + ~~DLK65~~	3	Background disc	29
DLIST	3	Background disc	31

QUEUE is provided with a priority of 2Ø which may be changed to meet system requirements as long as its priority remains higher (numerically lower) than the module TAM and lower (numerically higher) than the module DISP. It must be core-resident.

QUDIS is a privileged subroutine.

DISP is provided with a priority of 1Ø which may be changed to meet system requirements as long as its priority remains higher than the modules TAM and QUEUE.

TAM is provided with a priority of 3Ø which may be changed as long as it retains the same priority as the modules RFAM, DISC, and ERR.

RFAM is provided with a priority of 30, which may be changed as long as it retains the same priority as TAM, DISC, and ERR.

DISC is provided with a priority of 30, which may be changed as long as it retains the same priority as TAM, RFAM, and ERR.

ERR is provided with a priority of 30, which may be changed as long as it retains the same priority as TAM, RFAM, and DISC.

LSTEN is provided as a background program with a priority of 80, which can be changed to suit the specific system requirements of the user.

PROGL is provided with a priority of 29. ~~It calls subroutine DLK65, which is part of the HP 12771A Serial Interface Kit, or HP 12773A Modem Interface Kit.~~

DLIST is provided with a priority of 31.

PROGL and DLIST have no specific placement requirements. They can be placed in the user HP 9701 System as either background or foreground programs with varying effects on system throughput. Foreground core-resident is the most advantageous placement, but core limitations often make this impractical. If either or both programs must be made disc resident and no extended background operations are anticipated during the hours of communications activity, the HP 91701 software provides the best remote file access, if they are placed in the background. If extended background activity is anticipated, the best placement is foreground disc resident with priority lower than the module TAM.

The File Management Package is also configured into the RTE System during this phase. The various modules of FMP are numbered according to the order of their loading; other distinctions are as follows:

FMGR

The operator interface MFGR has a priority of 90, is segmented into five parts, and requires at least 5K of background area.

D.RTR

Subroutine D.RTR has a priority of one, requires a few words over 1K of area, and is supplied as a foreground disc resident program. However, to assure minimum response time, the HP 9701 System requires it be made core resident.

This change can be made during the parameter input phase of RTGEN. The only constraint is that D.RTR have a higher priority than any program using the EMP.

PARAMETER INPUT PHASE

During the parameter input phase, the operator can modify the type, priority or execution intervals of any of the programs entered during the program input phase. The program type code for background main programs and their segments cannot be changed without losing their relationship to each other.

Each parameter record is of this general form:

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

The types and priorities assembled in the NAM records of the HP 91701 software modules for 32K are shown in Table 6-1. These values can be used as references to make required changes. The overrides required for a 24K RTE central configuration are shown in Table 6-2.

Table 6-2. 24K RTGEN Parameter Overrides

Module Type	Type	Default Residency	Default Priority
D.RTR	1	Foreground core	1
QUEUE	1	Foreground core	20
DISC	2	Foreground disc	30
LSTEN	3	Background disc	90
QUDIS	6	Library	none
DISP	2	Foreground disc	10
TAM	2	Foreground disc	30
ERR	2	Foreground disc	30
RFAM	2	Foreground disc	30
PROGL	2	Foreground disc	~~35~~ 29
DLIST	2	Foreground disc	35

DISC LOADING PHASE

RTGEN generates three I/O tables during this phase: equipment table, device reference table, and the interrupt table. The material which follows indicates HP 91701 requirements for these tables.

Equipment Table Entries

One Equipment Table Entry (EQT) is required for each terminal in the system. The terminal computer I/O channels must be related to DVR65, and DMA must be specified. The format is:

nn, DVR65, D

where

nn = I/O channel number

Device Reference Table Entries

A unique logical unit number must be assigned to each terminal. The terminals are not required to be in contiguous logical unit sequence.

Interrupt Table Entries

An interrupt on any terminal I/O channel must schedule the module QUEUE. The option PRG is required. The format is

nn, PRG, QUEUE

where

nn is the I/O channel (select code)

When the RTGEN process is complete and the system booted up, FMGR must be initialized; the HP 91701 module LSTEN (used to initialize the communication link between central and the terminal computers) must be initialized to make the HP 91701 system fully operational.

FMGR Initialization

Each time the RTE System is loaded from the disc, the FMGR program is scheduled. The first time that FMGR is scheduled, it detects that the FMP has not been initialized to the system and auxiliary discs. Once this initialization procedure (described in the following steps) takes place, later RTE System loads will still schedule FMGR, but initialization is no longer necessary.

When FMGR is scheduled, it obtains all available tracks on the system and auxiliary discs, and assigns the tracks to itself. FMGR then prints on the system teleprinter (TTY):

```
FMGR ØØ2
  :
```

This is a request for the user to initialize the system disc (LU2) using the IN Operator Command.

NOTE

The security code entered at this time will be the system master security code. The code can be blank (no security) or any two characters (except a colon, comma, or leading blank). The two characters need not be printing characters. Remember, the code may not be obtained with any FMGR Command once it is set, so be sure to remember it.

After a successful initialization of the system disc, FMGR checks to see if there is an auxiliary disc. If so, FMGR prints on the system TTY:

```
FMGR ØØ3
:
```

This is a request for the user to initialize the auxiliary disc (LU3) using the IN Operator Command. If no tracks are to be assigned to the auxiliary disc, its label should be specified as zero. For example:

:IN,JB,-3,Ø

When a successful initialization is completed, FMGR assigns the tracks as per the IN command parameters. FMGR then terminates (no message is printed) and returns control to the RTE System.

If the initialization was not successful, that is, if any tracks were not available, FMGR prints:

FMGR ØØ5
FMGR *xxx*

ØØ5 identifies the message and *xxx* is the track's relative position in the track assignment table. This implies the track number is printed for the system disc, and the track number plus the number of tracks on the system disc is printed for the auxiliary disc. Recoveries from this error condition are:

a. Make the track available and then enter the following system commands:

*RT,D.RTR
*ON,FMGR

This will force the FMGR to re-try the track assignment table setup.

b. Re-initialize the affected disc declaring the unavailable tracks as bad (message will still be printed but may be ignored).

c. Re-initialize the affected disc changing the first track to be above the unavailable track.

If the unavailable track is the last system track, procedure "a" must be used; i.e., this track must be available to the FMP.

HP 91701 INITIALIZATION

The HP 91701 module, LSTEN, is used to initialize the communication link between the central computer of the Distributed System and its terminal computers. This is done by initializing the communication cards in both machines. LSTEN is scheduled with an ON,LSTEN request from the central computer. The format for this request is:

ON, LSTEN,*p1*,*p2*,*p3*

where

p1 = Logical unit number of the system console. The default value is the CRT (1).

p2 = Logical unit number of the paper tape reader. The default value is (5).

p3 = Logical unit number of the paper tape punch. The default value is (4).

Two commas in a row mean the parameter is not specified and the default is to be used.

If incorrect, but defined, LUs are supplied to any of the LSTEN questions, the following messages are displayed:

INCORRECT INPUT PARAMETERS
LSTEN ABORTED

If undefined LUs are supplied as responses, the following messages are displayed:

IOØ3 LSTEN *address*
LSTEN ABORTED

The LSTEN module provides two options: (1) initialize the communication links for the entire system, (2) reinitialize the communication links between the central computer and a single terminal. The first option is used for system start-up. The second option is used to reestablish terminal/central communication any time a terminal program destroys the communication link.

Option 1 requires that the logical unit numbers for all terminals in the communication network be provided. It also requires the input of the maximum number of files concurrently open to all terminals.

Option 2, to reinitialize the communication link between a single terminal and central requires the logical unit number of the terminal.

To initialize system communication links, the following sequence is performed at the central CRT:

OPTION

1 Standard option to initialize communication links for the system.

2 Reinitialize communication link for a single terminal. If an incorrect option is specified, the following messages are displayed:

INCORRECT INPUT PARAMETERS
OPTION?

OPTION 1

If the system has not been re-booted since the last time this option was used, the following error message is displayed:

RE-BOOT REQUIRED FOR THIS OPTION
IS INITIALIZATION INFO ON PAPER TAPE?

A syntax error in the response causes the question to be re-asked.

YES Read paper tape.

If YES and no tape:

I/O ERR TO EQT *xx*

Where *xx* is the LU number of the paper tape reader.

IOØ2 LSTEN *address*

LSTEN ABORTED

NO Continue to next statement

If the initialization information is not on paper tape, the following information must be supplied:

ENTER ALL TERMINAL LOGICAL UNIT #'s

Enter the logical unit numbers of all terminals to be initialized for communication. The numbers **must** be entered in one line, separated by commas and without intervening

(CR) / (LF) .

If incorrect parameters are entered, the question is re-asked.

If undefined logical unit numbers are input, an IOØ3 error results and LSTEN aborts.

ENTER MAXIMUM # OF TERMINAL OPEN FILES

Legal values are between 1 and 256.

If zero or a negative value is entered, the question is re-asked.

If a value greater than 256 is input, the following error message is displayed and the question is repeated.

INCORRECT INPUT PARAMETER

If there is not enough room on the disc for the overflow file, the following error message is displayed and LSTEN aborts.

DISC FULL

INITIALIZATION PAPER TAPE DESIRED?

YES Produces a paper tape containing logical unit numbers and maximum numbers of open files, which are entered instead of CRT inputs for a rerun of LSTEN. If the punch is off when this answer is entered, I/O ERROR ON EQT *xx* is displayed.

Where *xx* is the LU of the punch.

NO Continue to next statement.

A syntax error in the response causes the question to be re-asked.

9701 HAS BEEN INITIALIZED AND IS OPERATING

The initialization process is complete and the communication link enabled.

To reinitialize a terminal communication link, the following sequence is performed at the central system console:

OPTION 2

Reinitialize communications link for a single terminal.

TERMINAL LOGICAL UNIT #TO CLEAR?

If the logical unit numbers input is undefined, the following messages are displayed:

IOØ3 LSTEN *address*
LSTEN ABORTED

If the logical unit number is incorrect, but system defined, the following message is displayed:

INCORRECT INPUT PARAMETER

The operator must re-enter the parameter with the correct number.

TERMINAL CLEARED AND LSTEN MODE ENABLED

The reinitialization process is complete.

TERMINAL INITIALIZATION

The terminals are initialized as described in the first three parts of this section.

RTGEN SAMPLES

The RTGEN examples which follow are included to show how the 91701 software modules are configured into the Real-Time Executive System. The examples consist of two parts: (1) Configuration worksheets for RTGEN planning, and (2) the RTE generator output listing.

A complete description of the Real-Time Executive Generation process is contained in the **Real-Time Executive System Manual.**

The first example is for a 24K RTE central computer with moving head disc, having the following system configuration: 1 disc platter, 1 magnetic tape, 6 terminal computers (remote CPUs), 1 local CRT, 1 line printer (the HP 2767A Line Printer is driven by DVRØØ as if it were an HP 2605A), 1 paper tape punch, 1 photo reader, and 1 system teletype. This example also provides 1Ø extra ID segments for user programs. The core allocation map for this example is provided in Appendix B.

The configuration worksheets shown here are described in Part 1 of Section VI of the **Real-Time Executive System Manual.**

MOVING HEAD DRIVE	MOVING HEAD DRIVE	MOVING HEAD DRIVE	MOVING HEAD DRIVE
SUBCHANNEL 0	SUBCHANNEL 2	SUBCHANNEL 4	SUBCHANNEL 6
FIXED			
NO. OF TRACKS AVAILABLE, NO0 0	NO. OF TRACKS AVAILABLE, NO2 0	NO. OF TRACKS AVAILABLE, NO4 0	NO. OF TRACKS AVAILABLE, NO6 0
FIRST TRACK, FT0 0	FIRST TRACK, FT2 0	FIRST TRACK, FT4 0	FIRST TRACK, FT6 0
SUBCHANNEL 1	SUBCHANNEL 3	SUBCHANNEL 5	SUBCHANNEL 7
REMOVABLE			
NO. OF TRACKS AVAILABLE, NO1 200	NO. OF TRACKS AVAILABLE, NO3 0	NO. OF TRACKS AVAILABLE, NO5 0	NO. OF TRACKS AVAILABLE, NO7 0
FIRST TRACK, FT1 0	FIRST TRACK, FT3 0	FIRST TRACK, FT5 0	FIRST TRACK, FT7 0

SYSTEM SUBCHANNEL NUMBER Ø SCRATCH SUBCHANNEL NUMBER Ø

AUXILIARY SUBCHANNEL NUMBER NONE START SCRATCH (I.E., 1ST TRACK = 0) Ø

Figure 6-2. Moving Head Disc

INPUT/OUTPUT CONFIGURATION WORKSHEET

RTGEN NUMBER 24K DATE _____ PREPARED BY _____

STD. LOGICAL UNIT NOS.	I/O INTERFACE CARD NAME	P.I.	PRIVILEDGED SEP.	T.B.G.	TELETYPE	C.R.T.	PHOTO READER	PUNCH	LINE PRINTER	DISC (1)	DISC (2)			MAG TAPE (1)	MAG TAPE (2)		SDI Ø	SDI 1	SDI 2	SDI 3	SDI 4	SDI 5	< FUTURE APP >	< FUTURE APP >	< FUTURE APP >						
	SC / SUB	10	11	12	13	15	16	17	20	21	22	Ø	1	23	24	Ø	25	26	27	30	31	32									
1 SYS. TTY					X																										
2 SYS. MASS STORAGE													X																		
3 AUX. MASS STORAGE																															
4 PUNCH OUTPUT								X																							
5 INPUT							X																								
6 LIST OUTPUT									X																						
7_{10} TO 63_{10}					1	8	5	4	6				2			7	9	10	11	12	13	14									
DVR IDENT. (DVRxx)				7Ø	ØØ	ØØ	Ø1	Ø2	ØØ	31				23			65	65	65	65	65	65									
DMA REQUIRED (D)										D				D			D	D	D	D	D	D									
EQT ENTRY NO.				14	1	2	3	4	5	6				7			8	9	10	11	12	13	15	16	17						
BUFFERED OUTPUT (B)					B	B		B	B																						
TIME-OUT (T)					T	T	T		T	T	T			T																	
OCTAL SELECT CODE / SUBCHANNEL	10																														

Figure 6-3. Input/Output Configuration Worksheet for 24K RTE Central

```
MH. DISC CHNL?
23
# TRKS, FIRST TRK ON SUBCHNL.
0?
0 , 0
1?
200 , 0
2?
 , 
3?
 , 
4?
 , 
5?
 , 
6?
 , 
7?
 , 
SYSTEM SUBCHNL?
1
AUX DISC (YES OR NO)?
NO
AUX DISC SUBCHNL?

SCRATCH SUBCHNL?
0
START SCRATCH?
0
# 128 WORD SECTORS/TRACK?
48
TBG CHNL?
12
PRIV. INT. CARD ADDR.
0
SWAPPING?
YES

LWA MEM?
57677
PRGM INPT?
PT
LIBR INPT?
MT
PRAM INPT?
TY

INITIALIZE SUBCHNL.n
(0?)

(1?)

(2?)

(3?)

(4?)

(5?)

(6?)

(7?)

PUNCH BOOT?
YES
YES

NO
```

Figure 6-4. Initialization Phase – MH

```
NAME, TYPE [, PRIORITY] [, EXECUTION INTERVAL]

D.RTR , 1 , 1 ,
QUEUE , 1 , 20 ,
DISC , 2 , 30 ,
LSTEN , 3 , 90 ,
QUDIS , 6 , ,
DISP , 2 , 10 ,
TAM , 2 , 30 ,
ERR , 2 , 30 ,
BFAM , 2 , 30
PROGL , 2 , 35 29
DLIST , 2 , 35
FMGR , 3 , 90
ASMB , 3 , 95
ALGOL , 3 , 95
FTN4 , 3 , 95
EDIT , 3 , 96
LOADR , 3 , 93
SXL , 3 , 92

/E
```

Figure 6-5. Parameter Input Phase

```
* INTERRUPT TABLE
10 , ABS , 106710
11 , ABS , 106711
13 , EQT , 1
14 , ABS , 106714
15 , EQT , 12
16 , EQT , 13
17 , EQT , 4
20 , EQT , 5
21 , EQT , 6
22 , EQT , 6
23 , EQT , 7
24 , EQT , 7
25 , PRG , QUEUE
26 , PRG , QUEUE
27 , PRG , QUEUE
30 , PRG , QUEUE
31 , PRG , QUEUE
32 , PRG , QUEUE
```

```
* EQUIPMENT TABLE ENTRY
(1)  13 , DVR00 , B , T , , T = 9999
(2)  15 , DVR00 , B , T , , T = 9999
(3)  16 , DVR01 , T , , , T = 9999
(4)  17 , DVR02 , B , T , , T = 9999
(5)  20 , DVR00 , B , T , , T = 100
(6)  21 , DVR31 , D , T , , T = 9999
(7)  23 , DVR23 , D , T , , T = 9999
(8)  25 , DVR65 , D , , , T =
(9)  26 , DVR65 , D , , , T =
(10) 27 , DVR65 , D , , , T =
(11) 30 , DVR65 , D , , , T =
(12) 31 , DVR65 , D , , , T =
(13) 32 , DVR65 , D , , , T =
(14) 12 , DVR70 , , , , T =
```

Figure 6-6. Disc Loading Phase

```
                    *DEVICE REFERENCE TABLE

1 = EQT #? (SYSTEM TELEPRINTER)          17 = EQT #?
1 , _____                                0 , _____

2 = EQT #? (SYSTEM MASS STORAGE)         18 = EQT #?
6 , _____                                0 , _____

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

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

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

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

7 = EQT #?
7 , _____

8 = EQT #?
2 , _____

9 = EQT #?
8 , _____

10 = EQT #?
9 , _____

11 = EQT #?
10 , _____

12 = EQT #?
11 , _____

13 = EQT #?
12 , _____

14 = EQT #?
13 , _____

15 = EQT #?
14 , _____

16 = EQT #?
0 , _____
```

Figure 6-6. Disc Loading Phase (Continued)

INITIALIZATION PHASE

```
MH DISC CHNL?
21
```

RTGEN requests the higher priority select code (octal) of the system disc controller.

```
# TRKS, FIRST TRK ON SUBCHNL:
  07
 0
   1?
 200,0
   2?
 /E
```

Starting track and number of tracks (decimal) of each subchannel assigned within this system.

```
START SCRATCH?
0
```

Relative track number upon which to start disc scratch area.

```
# 128 WORD SECTORS/TRACK?
48
```

Number of 128-word sectors (decimal) per logical track

```
TBG CHNL?
12
```

Select code for the time base generator (octal).

```
PRIV. INT. CARD ADDR?
0
```

Address of the privileged interrupt I/O card, if present.

```
SWAPPING?
YES
```

Real-time disc resident programs can swap in and out of core according to priority .

```
LWA MEM?
57677
```

The last word of available core memory in octal (77677_8 for a 32K computer).

```
PRGM INPT?
PT
```

Input unit for relocatable program modules (PT – paper tape, TY – teleprinter, MT – magnetic tape, or DF – disc file).

```
LIBR INPT?
MT
```

Input unit for relocatable library programs (same choices as above).

```
PRAM INPT?
TY
```

Input unit for parameters describing relocatable programs (PT or TY).

There is a pause here while RTE initializes the system disc.

```
PUNCH BOOT?
YES

PUNCH BOOT?
YES

PUNCH BOOT?
NO
```

Paper Tape bootstrap punch option. The first time the tape bootstrap is indicated, the answer should be Yes. One copy of the bootstrap tape is loaded and run from 100_8 to bring the finished system up.

```
ERR 08
FUSS
ERR 05
SMBL
*EOT
```

This output results when the RTE Module EXEC is replaced by the 91701 Module EXEC. These errors have no operational effects.

PROGRAM INPUT PHASE

A halt 77 occurs here.

The switch register is set = 2 (for paper tape library input) and the 9701 programs are loaded as follows:

A. Put LSTEN in paper tape reader, press RUN
B. Put QUDIS in paper tape reader, press RUN
C. Put QUEUE in paper tape reader, press RUN
D. Put DISP in paper tape reader, press RUN
E. Put ERR in paper tape reader, press RUN
F. Put TAM in paper tape reader, press RUN
G. Put RFAM in paper tape reader, press RUN
H. Put DISC in paper tape reader, press RUN

An EOT (end of tape) is output after each program is loaded.

The system library is mounted and the switch register cleared if on magnetic tape or set equal to 2 for paper tape. When run is pressed, the library is loaded. This operation takes some time and is followed by the EOT message.

After the EOT, the switch register is set to 1, and RUN is pressed. This causes RTGEN to print all undefined externals.

```
NO UNDEF EXTS
```

No undefined externals indicates that all is OK. Press RUN to continue.

PARAMETER INPUT PHASE

```
PARAMETERS

D,RTR,1,1
QUEUE,1,20
DISC,2,30
LSTEN,3,90
QUDIS,6
DISP,2,10
TAM,2,30
ERR,2,30
RFAM,2,30
PROGL,2,35
DLIST,2,35
FMGR,3,90
ASMB,3,95
ALGOL,3,95
FTN4,3,95
EDIT,3,96
LOADR,3,93
SXL,3,92
/E
```

Operator modifies type, priority, or execution intervals of any programs entered during program input phase (except background main types).

Each parameter record is of the general form:

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

```
# OF BLANK ID SEGMENTS?
10
```

Number of blank ID segments to be allocated for on-line loading of programs by the relocating loader.

DISC LOADING PHASE

```
FWA BP LINKAGE?
33
```

First word of available core memory in base page (first available octal select code number after last I/O card).

```
SYSTEM

EXEC (00)   02000

SCHED(00)   03570

RTIOC(00)   10762

DVR00(00)   14043

DVR23(00)   15207

DVR31(00)   16012

DVR65(00)   17203
  LT,65     21075

DVR70(00)   21076
```

RTGEN OUTPUT of memory map beginning with the modules of the Real-Time Executive.

```
F4D,B(00)  21102

F2E,D(00)  21102

BP LINKAGE 00513

* EQUIPMENT TABLE ENTRY
```

RTGEN begins generating the equipment table from user information.

```
13,DVR00,B,T
 T =
9999
```

TTY driver

```
15,DVR00,B,T
 T =
9999
```

TTY driver

```
16,DVR01,T
 T =
9999
```

Photo reader

```
17,DVR02,B,T
 T =
9999
```

Paper tape punch

```
20,DVR00,B,T
 T =
100
```

TTY driver

```
21,DVR31,D,T
 T =
9999
```

Moving head disc driver

```
23,DVR23,D,T
 T =
9999
```

Magnetic tape driver

```
25,DVR65,D

26,DVR65,D

27,DVR65,D
```

Communications serial interface driver (one for each I/O Board)

```
30,DVR65,D

31,DVR65,D

32,DVR65,D

12,DVR70
```

Dummy driver

```
/E
```

```
* DEVICE REFERENCE TABLE

 1 = EQT #?
1

 2 = EQT #?
6,1

 3 = EQT #?
0,0

 4 = EQT #?
4

 5 = EQT #?
3

 6 = EQT #?
5

 7 = EQT #?
7

 8 = EQT #?
2

 9 = EQT #?
8

10 = EQT #?
9

11 = EQT #?
10

12 = EQT #?
11

13 = EQT #?
12

14 = EQT #?
13

15 = EQT #?
14

16 = EQT #?
0

17 = EQT #?
0
```

Logical Unit numbers 1 through 6 are predefined in the RTE System as:

1 - System teleprinter
2 - System mass storage
3 - Auxiliary mass storage
4 - Standard punch unit
5 -Standard input unit
6 - Standard list unit

```
* INTERRUPT TABLE
10,ABS,106710
11,ABS,106711
13,EQT,1
14,ABS,106714
15,EQT,12
16,EQT,13
17,EQT,4
20,EQT,5
21,EQT,6
22,EQT,6
23,EQT,7
24,EQT,7
25,PRG,QUEUE
26,PRG,QUEUE
27,PRG,QUEUE
30,PRG,QUEUE
31,PRG,QUEUE
32,PRG,QUEUE
/E
```

Interrupt table generation requires an entry for each I/O card in ascending order.

Interrupt table complete.

```
LIBRARY
  QUDIS     26373
  .ENTR     27075
  PRTN      27165
BP LINKAGE 00525
RT RESIDENTS
QUEUE(20)  27270
```

RTGEN outputs the names and entry point addresses of all library routines which are referenced by Real-Time and background programs.

New first word of available core memory in base page.

RTGEN begins loading the Real-Time disc-resident programs.

```
D.RTR(01)   30111
  P.PAS     32003

BP LINKAGE  00543

RT DISC RESIDENTS

DISP (10)   32032

TAM  (30)   32032

RFAM (30)   32032
  R/W$      37743
  RMPAR     40022
  GETAD     40045

DISC (30)   32032
  RMPAR     33721
  NAMF      33744
  FCONT     34116
  LOCF      34210
  APOSN     34370
  CREAT     34507
  PURGE     34766
  OPEN      35065
  READF     35250
  POSNT     36035
  RWNDF     36272
  CLOSE     36354
  FSTAT     36463
  GETAD     36510
  NAM..     36526
  P.PAS     36623
  RW$UB     36652
  $OPEN     37120
  R/W$      37275
  RWND$     37354

ERR  (30)   32032
  RMPAR     32153
  GETAD     32176

PROGL(35)   32032
  .EAU.     33175
  CLRIO     33245
  IAND      33250
  RMPAR     33260
  OPEN      33303
  READF     33466
  DLK65     34253
  CLOSE     34325
  GETAD     34434
```

RTGEN begins loading the Real-Time disc-resident programs.

```
  $OPEN     34452
  R/W$      34627
  P.PAS     34706
  RW$UB     34735
  RWND$     35203

DLIST(35)   32032
  FSTAT     33312

BP LINKAGE 00647

CHANGE BP LINKAGE?
1100

FWA SY MEM 40063

CHANGE FWA SYS AV MEM?
0

BG BOUNDRY?
42000

BG RESIDENTS

(NONE)

BG DISC RESIDENTS

ASMB (95)   42000

ASMBD(99)   46643

ASMB1(99)   46643

ASMB2(99)   46643

ASMB3(99)   46643

ASMB4(99)   46643

ASMB5(99)   46643

ALGOL(95)   42000
  .EAU.     54312
  %WRIT     54362
  $READ     55063
  .OPSY     55621

ALGL1(99)   55661

FTN4 (95)   42000
```

Report of new first word of available core memory in base page.

Linkage area is changed for future addition of larger Real-Time disc-resident programs on-line.

First word address of system available memory.

An address change would increase the Real-Time disc-resident area for future addition of larger programs.

First address of the background area (area from 47000_8 to 50000_8 is used for temporary storage of output buffers, and reentrant temporaries).

Begin load of background disc-resident programs, with names and entry points for mains and subroutines.

```
FTN40(99)   50311

FTN41(99)   50311

FTN42(99)   50311

FTN43(99)   50311

FTN44(99)   50311

FTN45(99)   50311

FTN46(99)   50311

FTN47(99)   50311

FTN48(99)   50311

FTN49(99)   50311

FTN4A(99)   50311

FTN4B(99)   50311

FTN4C(99)   50311

FTN4D(99)   50311

FTN4E(99)   50311

FTN4F(99)   50311

FTN4G(99)   50311

FTN4H(99)   50311

EDIT (96)   42000
  SREAD     44411
  %WRIS     45147

  .OPSY     45437

LOADR(93)   42000

SXL  (92)   42000
  SGMTR     42047
  OPROC     42134
  LDUMM     42325
  DBUGS     43367
  .EAU.     44505
  SPTRU     44555
  DTSTR     45434
```

```
   PRFOP     45577
   FU5S      46251
   JSBES     46317
   ULIBR     46324
   ALLOC     46372
   PUT4      46560
   DIAG      46630
   PAGER     46654
   .DRCT     47737

SG01L(99)    47746
   SCNSM     50056
   SCNPT     50761
   LCRST     51160
   SCNMN     51633
   FU5T      52024
   FU5V      52057
   NXSY      52123
   SNAM      52210
   MAT1      52437
   MAXR      52600
   PARSE     52652
   SCGRM     53125

SG02L(99)    47746
   SNPST     50056
   LDUM2     50255
   SNTX      50461
   UGRMR     50737
   FKDCB     51727
   FU5T      52011
   FU5V      52044
   FSEMU     52110
   PARSE     52245
   FU9       52520

SG03L(99)    47746
   ENDR      50056
   SWTCH     51047
   BLNK      51110
   MOVE.     51126
   POSN.     51145
   RVERF     51212
   OCTAQ     51456
   MAXR      51546
   PDLG      51620
   REDGL     52015
   .LOCF     52153
   READF     52177
   R/WS      52764
   .XEC      53043
   RDDSK     53113
```

```
   P.PAS     53241
   RWSUB     53270
   RWND$     53536

SG04L(99)    47746
   ENDR2     50056
   .XEC      50476
   MSTBL     50546
   PRENT     50650
   CFXUP     51012
   BLOK      51413
   CCON      51547
   STFM      51572
   SNAM      51617
   FU5T      52046
   FU5V      52101
   FU5A      52145
   BLNK      52210
   OCTAQ     52226
   MAXR      52316
   PBPLK     52370
   ABOUT     52444
   NXSY      52545
   STMA      52632
   READF     53022
   R/W$      53607
   P.PAS     53666
   RWSUB     53715
   RWND$     54163

SG05L(99)    47746
   LDUM4     50056
   RELST     50346
   TERMF     50642
   FU5V      51670
   MSTBL     51734
   TRBAK     52036
   BLNK      52201
   OCTAQ     52217
   CLOSE     52307
   NXSY      52416
   FU5T      52503
   OPEN      52536
   APOSN     52721
   FKDCB     53040
   MOVE.     53122
   R/W$      53141
   RMPAR     53220
   $OPEN     53243
   RWSUB     53420
   LOCF      53666
   GETAD     54046
```

```
  RWNDS     54064
  P.PAS     54174

SG06L(99)   47746
  DGLST     50056
  FU5V      51515
  NXSY      51561
  CCON      51646
  MSTBL     51671
  .XEC      51773
  .GOTO     52043
  UNSTR     52066
  OCTAO     52170
  WACT8     52260
  MOVE.     52434
  LOCF      52453
  CLOSE     52633
  FKDCB     52742
  DIVD      53024
  P.PAS     53042
  R/WS      53071

SG07L(99)   47746
  LOUM4     50056
  XFRST     50346
  FU5V      50613
  MSTBL     50657
  FU5T      50761
  TRBAK     51014
  OPEN      51157
  ILOSE     51342
  LOCF      51365
  FKDCB     51545
  MOVE.     51627
  BLNK      51646
  APOSN     51664
  CLOSE     52003
  RMPAR     52112
  SOPEN     52135
  P.PAS     52312
  RWSUB     52341
  R/WS      52607
  GETAD     52666
  RWNDS     52704

SG08L(99)   47746
  DSPST     50056
  DSUDF     50436
  OCTAO     50662
  MSTBL     50742
  BLNK      51044
  FU5V      51062
```

```
   FU5T      51126
   NXSY      51161
   CFXUP     51246
   UNSTR     51647
   CCON      51751
   MAXR      51774
   PBPLK     52046
   ABOUT     52122
   READF     52223
   R/WS      53010
   P.PAS     53067
   RWSUB     53116
   RWNDS     53364

SG09L(99)    47746
   SETSM     50056
   FU5T      50600
   FU5V      50633
   STMA      50677
   ABOUT     51067
   PRENT     51170
   CFXUP     51332
   BLNK      51733
   READF     51751
   MAXR      52536
   MSTBL     52610
   OCTAQ     52712
   PBPLK     53002
   R/WS      53056
   P.PAS     53135
   RWSUB     53164
   RWNDS     53432

SG10L(99)    47746
   OTPT      50056
   READF     51632
   OPEN      52417
   CLOSE     52602
   MSTBL     52711
   ABOUT     53013
   MOVE.     53114
   BLNK      53133
   FU5V      53151
   R/WS      53215
   P.PAS     53274
   RWSUB     53323
   RMPAR     53571
   SOPEN     53614
   MAXR      53771
   RWNDS     54043
   GETAD     54153
```

```
SG11L(99)   47746
  NAMR      50056
  SCOMM     50635
  .LOCF     50736
  FU5V      50762
  GNNS      51026
  MAXR      51077
  STMA      51151
  SWTCH     51341
  RVERF     51402
  MOVE.     51646
  GET4      51665
  POLG      51721
  REDGL     52116
  POSN.     52254
  READF     52321
  .XEC      53106
  RDDSK     53156
  R/W$      53304
  P.PAS     53363
  RWSUB     53412
  RWND$     53660

SG12L(99)   47746
  LOD1      50056
  SKIPR     50355
  FUST      50544
  STFM      50577
  GNNS      50624
  STMA      50675
  MAT1      51065
  RVERF     51226
  SWTCH     51472
  MOVE.     51533
  NXSY      51552
  GET4      51637
  .XEC      51673
  POLG      51743
  REDGL     52140
  POSN.     52276
  .LOCF     52343
  READF     52367
  RDDSK     53154
  R/W$      53302
  P.PAS     53361
  RWSUB     53410
  RWND$     53656

SG13L(99)   47746
  OTPST     50056
  LDM12     50416
  FUST      50671
  FU5V      50724
```

```
   FKDCB     50770
   CREAT     51052
   MSTBL     51331
   OPEN      51433
   BLNK      51616
   CLOSE     51634
   $OPEN     51743
   NAM..     52120
   RMPAR     52215
   R/W$      52240
   RWND$     52317
   GETAD     52427
   RW$UB     52445
   P.PAS     52713

SG14L(99)    47746
   INIT2     50056
   AEND      50137
   CRDBO     50372
   DSUDF     50541
   MSTBL     50755
   CREAT     51057
   .XEC      51336
   MOVE.     51406
   CFXUP     51425
   UNSTR     52026
   BLNK      52130
   CCON      52146
   CLOSE     52171
   $OPEN     52300
   NAM..     52455
   RMPAR     52552
   MAXR      52575
   PBPLK     52647
   ABOUT     52723
   R/W$      53024
   RWND$     53103
   GETAD     53213
   READF     53231
   RW$UB     54016
   P.PAS     54264

SG15L(99)    47746
   CLOP      50056
   PKUDF     50073
   CLOSE     50157
   R/W$      50266

SG16L(99)    47746
   UBL       50056
   ABOUT     51061
   SWTCH     51162
```

```
   RVERF     51223
   FU5V      51467
   MAXR      51533
   LLINK     51605
   READF     52215
   POLG      53002
   REDGL     53177
   POSN.     53335
   .LOCF     53402
   .XEC      53426
   PBPLK     53476
   R/W5      53552
   P.PAS     53631
   RWSUB     53660
   RDDSK     54126
   RWND$     54254

SG17L(99)    47746
   RDCRD     50056
   READF     50304
   TRBAK     51071
   R/W$      51234
   P.PAS     51313
   RWSUB     51342
   OPEN      51610
   APOSN     51773
   FKDCB     52112
   MOVE.     52174
   RWND$     52213
   CLOSE     52323
   RMPAR     52432
   $OPEN     52455
   LOCF      52632
   GETAD     53012
SG18L(99)    47746
   CRTOP     50056
   FU5V      50201
   BLNK      50245
   MSTBL     50263
   CREAT     50365
   CLOSE     50644
   $OPEN     50753
   NAM..     51130
   RMPAR     51225
   R/W$      51250
   RWND$     51327
   GETAD     51437
   RWSUB     51455
   P.PAS     51723

SG19L(99)    47746
   SNPOP     50056
```

```
   CCON      51367
   NXSY      51412
   FUSV      51477
   FUST      51543
   UNSTR     51576
   READF     51700
   MSTBL     52465
   OCTAQ     52567
   ABOUT     52657
   BLNK      52760
   R/WS      52776
   P.PAS     53055
   RWSUB     53104
   MAXR      53352
   RWND$     53424

SG20L(99)    47746
   LNKST     50056
   MAPST     50137
   SGINT     50501
   FUSV      50671
   DINIT     50735
   INIT1     51213
   .XEC      51417
   MAXR      51467
   MOVE.     51541
   SYM       51560
   MEMR      52361
   OPEN      52422
   FKDCB     52605
   CLOSE     52667
   RMPAR     52776
   $OPEN     53021
   R/W$      53176
   GETAD     53255
   RWND$     53273
   RWSUB     53403
   P.PAS     53651

SG21L(99)    47746
   OPLG      50056
   SNAM      50637
   FUST      51066
   FUSV      51121
   FUSA      51165
   CFXUP     51230
   PRENT     51631
   MAT2      51773
   CCON      52115
   BLNK      52140
   MSTBL     52156
   MAXR      52260
```

```
   PBPLK     52332
   ABOUT     52406
   OCTAQ     52507
   READF     52577
   R/W$      53364
   P.PAS     53443
   RWSUB     53472
   RWND$     53740

FMGR (90)    42000
   FM.CM     42643
   .EAU.     44451
   .DRCT     44521
   READF     44530
   CLOSE     45315
   OPEN      45424
   R/W$      45607
   P.PAS     45666
   RWSUB     45715
   RMPAR     46163
   $OPEN     46206
   RWND$     46363
   GETAD     46473

FMGR0(99)    46511
   PK..      46517
   CR..      50145
   CN..      51251
   FM.UT     51313
   LOCK.     52436
   RWNDF     52506
   ID.A      52570
   CREA.     52710
   NAM..     52762
   NAMF      53057
   CREAT     53231

FMGR1(99)    46511
   .PARS     46537
   REA.C     47453
   C.TAB     47536
   EE..      47617
   TR..      47654
   SA..      50045
   MR..      51007
   CREA.     51146
   LOCF      51220
   READ.     51400
   RWNDF     51425
   CK.SM     51507
```

```
   WRLG      51623
   CREAT     51641
   SREAD     52120
   %WRIT     52656
   NAM..     53357
   .OPSY     53454

FMGR2(99)    46511
   IN.IT     46520
   IN..      47372
   MC..      51210
   RC..      51520
   FM.UT     51707
   IPUT      53032
   FIU.      53053
   NAM..     53173
   J.PUT     53270
   LOCK.     53315
   MSC.      53365

FMGR3(99)    46511
   LI..      46517
   OL..      50230
   PU..      51453
   FM.UT     51676
   LOCF      53021
   MSC.      53201
   LOCK.     53236
   PURGE     53306

FMGR4(99)    46511
   ST.DU     46520
   CO..      47724
   SP..      50431
   CREA.     51206
   LOCF      51260
   CK.SM     51440
   FM.UT     51554
   CREAT     52677
   RWNDF     53156
   ID.A      53240
   NAM..     53360

FMGR5(99)    46511
   ??..      46523
   MS..      50564
   KP..      51063
   CL..      51705
   F.UTM     52167
   WRISS     52337
   IPUT      52376
   ID.A      52417
```

```
   LOCF      52537
   FSTAT     52717
   %WRIS     52744

 LSTEN(90)   42000
   CREAT     44534
   CLOSE     45013
   RMPAR     45122
   PURGE     45145
   SOPEN     45244
   NAM..     45421
   R/WS      45516
   GETAD     45575
   OPEN      45613
   RWNDS     45776
   RWSUB     46106
   P.PAS     46354

BP LINKAGE 01557

SYSTEM STORED ON DISC
SYS SIZE: 26 TRKS, 029 SECS(10)
```

The second RTGEN sample is for a 32K RTE central computer with moving head disc. The system configuration for this example is: 8 disc platters, 2 magnetic tapes, 8 remote CPUs (terminal computers), 1 local CRT, 1 line printer (the HP 2767A is driven by DVRØØ as if it were an HP 2605A), 1 paper tape punch, 1 photo reader, and 1 system teletype.

The core allocation map for this example is shown in Appendix B. The configuration worksheets which follow are in the same order as the RTGEN Planning discussion in the **Real-Time Executive Software System Manual.**

INPUT/OUTPUT CONFIGURATION WORKSHEET

RTGEN NUMBER _____ DATE _____ PREPARED BY _____

STD. LOGICAL UNIT NOS.	I/O INTERFACE CARD NAME	PRIVILEDGE INTERRUPT	PRIVILEDGE INTERRUPT	TIME BASE GENERATOR	TELETYPE	JUMPER	CRT	PHOTO READER	PUNCH	LINE PRINTER	DISC (1)	DISC (2)									MAG TAPE (1)	MAG TAPE (2)			SDI Ø	SDI 1	SDI 2	SDI 3	SDI 4	SDI 5	SDI 6	SDI 7
	SC / SUB	10	11	12	13	14	15	16	17	20	21	22	0	1	2	3	4	5	6	7	23	24	0	1	25	26	27	30	31	32	33	34
1	SYS. TTY				X																											
2	SYS. MASS STORAGE													X																		
3	AUX. MASS STORAGE																															
4	PUNCH OUTPUT								X																							
5	INPUT							X																								
6	LIST OUTPUT									X																						
7_{10} TO 63_{10}				11	1	9	10	5	4	6				2									7	8	12	13	14	15	16	17	20	21
DVR IDENT. (DVRxx)				70							31										23				65	65	65	65	65	65	65	65
DMA REQUIRED (D)											D										D				D	D	D	D	D	D	D	D
EQT ENTRY NO.				6	2		3	4	5	6	1										7				8	9	10	11	12	13	14	15
BUFFERED OUTPUT (B)					B		B		B	B	B																					
TIME-OUT (T)					T		T	T	T	T	T										T											
OCTAL SELECT CODE / SUBCHANNEL	10																															

Figure 6-7. Input/Output Configuration Worksheet for 32K RTE Central

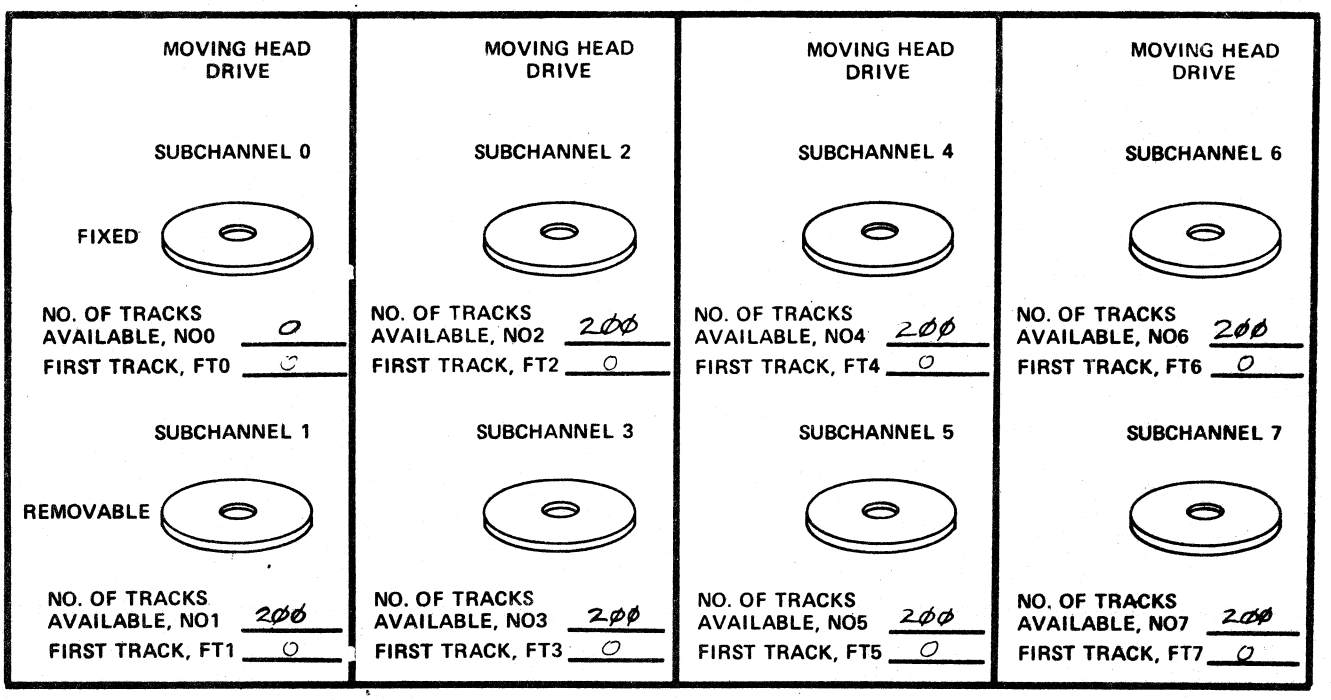


Figure 6-8. Moving Head Disc

```
MH. DISC CHNL?
21
# TRKS, FIRST TRK ON SUBCHNL.
Ø 0?, 0
200 1?, 0
200 2?, 0
200 3?, 0
200 4?, 0
200 5?, 0
200 6?, 0
200 7?, 0
SYSTEM SUBCHNL?
1
AUX DISC (YES OR NO)?
NO
AUX DISC SUBCHNL?

SCRATCH SUBCHNL?
1
START SCRATCH?
0
# 128 WORD SECTORS/TRACK?
48
TBG CHNL?
12
PRIV. INT. CARD ADDR.
0
SWAPPING?
YES
```

```
LWA MEM?
77677
PRGM INPT?
PT
LIBR INPT?
MT
PRAM INPT?
TY

INITIALIZE SUBCHNL.n
(0?)

(1?)

(2?)
NO
(3?)
NO
(4?)
NO
(5?)
NO
(6?)
NO
(7?)
NO
PUNCH BOOT?
YES

YES

NO
```

Figure 6-9. 32K Initialization Phase – MH

```
NAME, TYPE [,PRIORITY][,EXECUTION INTERVAL]

DRTR , 1 , 1 , ____ , ____ , ____ , ____ , ____ , ____
QUEUE , 1 , 20 , ____ , ____ , ____ , ____ , ____ , ____
QUEUE , 6 , ____ , ____ , ____ , ____ , ____ , ____ , ____
DISP , 1 , 10 , ____ , ____ , ____ , ____ , ____ , ____
IAM , 1 , 30 , ____ , ____ , ____ , ____ , ____ , ____
REAM , 2 , 30 , ____ , ____ , ____ , ____ , ____ , ____
DISC , 2 , 30 , ____ , ____ , ____ , ____ , ____ , ____
ERR , 1 , 30 , ____ , ____ , ____ , ____ , ____ , ____
SSTEN , 3 , 80 , ____ , ____ , ____ , ____ , ____ , ____
PROGL , 3 , 29 , ____ , ____ , ____ , ____ , ____ , ____
DLIST , 3 , 31 , ____ , ____ , ____ , ____ , ____ , ____

/E
# OF BLANK ID SEGMENTS?
 40
FWA BP LINKAGE
 35
```

Figure 6-10. 32K Parameter Input Phase

```
 * INTERRUPT TABLE
13 , EQT , 2
15 , EQT , 4
16 , EQT , 5
17 , EQT , 6
20 , EQT , 7
21 , EQT , 1
22 , EQT , 1
23 , EQT , 8
24 , EQT , 8
25 , PRG , QUEUE
26 , PRG , QUEUE
27 , PRG , QUEUE
30 , PRG , QUEUE
31 , PRG , QUEUE
32 , PRG , QUEUE
33 , PRG , QUEUE
34 , PRG , QUEUE
/E
```

```
 * EQUIPMENT TABLE ENTRY
(1)  21 , DVR31 , D , T ,      T = 9999
(2)  13 , DVR00 , B , T ,      T = 9999
(3)  15 , DVR00 , B , T ,      T = 9999
(4)  16 , DVR01 , T ,   ,      T = 9999
(5)  17 , DVR02 , B , T ,      T = 9999
(6)  20 , DVR00 , B , T ,      T = 100
(7)  23 , DVR23 , D , T ,      T = 9999
(8)  25 , DVR65 , D ,   ,      T =
(9)  26 , DVR65 , D ,   ,      T =
(10) 27 , DVR65 , D ,   ,      T =
(11) 30 , DVR65 , D ,   ,      T =
(12) 31 , DVR65 , D ,   ,      T =
(13) 32 , DVR65 , D ,   ,      T =
(14) 33 , DVR65 , D ,   ,      T =
(15) 34 , DVR65 , D ,   ,      T =
(16) 12 , DVR70 ,   ,   ,      T =
/E
```

```
/E
--------------------

     BP LINKAGE
     CHANGE BP LINKAGE?
X =  1100
     FWA SY MEM
     CHANGE FWA AV MEM?
Y =  45000
     BG BOUNDARY?
Z =  47000
     SYSTEM STORED ON DISC
     TRACK___SECTOR___(10)
```

Figure 6-11. 32K Disc Loading Phase

```
* DEVICE REFERENCE TABLE

 1 = EQT = ?  (SYSTEM TELEPRINTER)
 2 = EQT = ?  (SYSTEM MASS STORAGE)
 3 = EQT = ?  AUXILIARY MASS STORAGE)
 4 = EQT = ?  (STANDARD PUNCH UNIT)
 5 = EQT = ?  (STANDARD INPUT UNIT)
 6 = EQT = ?  (STANDARD LIST UNIT)
 7 = EQT = ?
 8 = EQT = ?
 9 = EQT = ?
10 = EQT = ?
11 = EQT = ?
12 = EQT = ?
13 = EQT = ?
14 = EQT = ?
15 = EQT = ?
11 , _____

16 = EQT = ?
12 , _____
17 = EQT = ?
13 , _____
18 = EQT = ?
14 , _____
19 = EQT = ?
15 , _____
20 = EQT = ?
 1 , 0
21 = EQT = ?
 1 , 2
22 = EQT = ?
 1 , 3
23 = EQT = ?
 1 , 4
24 = EQT = ?
 1 , 5
25 = EQT = ?
 1 , 6
26 = EQT = ?
 1 , 7
27 = EQT = ?
 0 , _____
28 = EQT = ?
 0 , _____
29 = EQT = ?
 0 , _____
/E
```

Figure 6-11. 32K Disc Loading Phase (Continued)

INITIALIZATION PHASE

```
HP DISC CHNL?
21
```
RTGEN requests the higher priority select code (octal) of the system disc controller.

```
# TRKS, FIRST TRK ON SUBCHNL
  0?
0
  1?
200,0
  2?
200,0
  3?
200,0
  4?
200,0
  5?
200,0
  6?
200,0
  7?
200,0
```
Starting track and number of tracks (decimal) of each subchannel assigned within this system.

```
SYSTEM SUBCHNL?
1
```
Subchannel number of system disc (LU2) that absolute code will be stored on.

```
AUX DISC (YES OR NO)?
NO
```
Auxiliary disc (LU3) not assigned.

```
SCRATCH SUBCHANL?
1
```
Subchannel number of scratch disc. This area is required for the relocatable modules used to build the system.

```
START SCRATCH?
0
```
Relative track number upon which to start disc scratch area.

```
# 128 WORD SECTORS/TRACK?
48
```
Number of 128-word sectors (decimal) per logical track.

```
TBG CHNL?
12
```
Select code for the time base generator (octal).

```
PRIV. INT. CARD ADDR?
0
```
Address of the privileged interrupt I/O card, if present.

```
SWAPPING?
YES
```
Real-time disc-resident programs can swap in and out of core according to priority.

```
LWA MEM?
77677
```
The last word of available core memory in octal (77677_8 for a 32K computer).

```
PRGM INPT?
PT
```
Input unit for relocatable program modules (PT – paper tape, TY – teleprinter, MT – magnetic tape, or DF – disc file).

```
LIBR INPT?
MT
```
Input unit for relocatable library programs (same choices as above).

6-52

```
PRAM INPT?
TY
```
Input unit for parameters describing relocatable programs (PT or TY).

There is a pause here while RTE initializes the system disc.

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

Initialize subchannels in addition to system, auxiliary and peripherals.

```
PUNCH BOOT?
YES

PUNCH BOOT?
YES

PUNCH BOOT?
NO
```

Paper tape bootstrap punch option. The first time the tape bootstrap is indicated, the answer should be Yes. One copy of the bootstrap tape is loaded and run from 100_8 to bring the finished system up.

```
ERR 26
FUSS
ERR 05
SMBL
*EOT
```

These error outputs result from replacing the RTE Module EXEC with the 91701 Module EXEC. The error output has no operational effect.

PROGRAM INPUT PHASE

A halt 77 occurs here.
The switch register is set = 2 (for paper tape library input) and the 9701 programs are loaded as follows:

A. Put LSTEN in paper tape reader, press RUN
B. Put QUDIS in paper tape reader, press RUN
C. Put QUEUE in paper tape reader, press RUN
D. Put DISP in paper tape reader, press RUN
E. Put ERR in paper tape reader, press RUN
F. Put TAM in paper tape reader, press RUN
G. Put RFAM in paper tape reader, press RUN
H. Put DISC in paper tape reader, press RUN

An EOT (end of tape) is output after each program is loaded.

The system library is mounted and the switch register cleared if on magnetic tape or set equal to 2 for paper tape. When run is pressed, the library is loaded. This operation takes some time and is followed by the EOT message.

After the EOT, the switch register is set to 1, and RUN is pressed. This causes RTGEN to print all undefined externals.

```
NO UNDEF EXTS
```

No undefined externals indicates that all is OK. Press run to continue.

PARAMETER INPUT PHASE

```
PARAMETERS

D.RTR,1,1
QUEUE,1,2
QUDIS,5
DISP,1,10
TAM,1,30
RFAM,2,50
DISC,2,90
ERR,1,30
LSTEN,3,50
PROGL,3,29
DLIST,3,31
/E
```

Operator modifies type, priority, or execution intervals of any programs entered during program input phase (except background main types).

Each parameter record is of the general form:

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

```
# OF BLANK ID SEGMENTS?
40
```

Number of blank ID segments to be allocated for on-line loading of programs by the relocating loader.

DISC LOADING PHASE

```
FWA BP LINKAGE?
35
```

First word of available core memory in base page (first available octal select code number after last I/O card).

```
SYSTEM
```

```
EXEC (PA)   02000
```

RTGEN OUTPUT of memory map beginning with the modules of the Real-Time Executive.

```
SCHED(PA)   03570

RTIOC(PB)   10762

DVR20(P1)   14043

DVR23(P1)   15207

DVR31(P1)   16012

DVR65(PU)   17203
  LT.65     21075

DVR70(PV)   21076

F4D.R(PB)   21102

F2E.R(PB)   21102
```

```
BP LINKAGE 00515
```

New first word of available core memory in base page.

```
* EQUIPMENT TABLE ENTRY
```

RTGEN begins generating the equipment table from user information.

```
21,DVR31,B,1
 T =
9999

13,DVR00,B,T
 T =
9999

15,DVR05,B,1
 T =
9999
```

TTY driver

```
16,DVR01,T
 T =
9999
```

Photo reader

```
17,DVR02,B,T
 T =
9999
```

Paper tape punch

```
20,DVR00,B,T
 T =
100
```

Line printer driver

```
23,DVR23,B,1
 T =
9999
```

Magnetic tape driver

```
25,DVR65,B
```

Communications serial interface driver (one for each IO board)

```
26,DVR65,B

27,DVR65,B

30,DVR65,B

31,DVR65,B

32,DVR65,B

33,DVR65,B

34,DVR65,B

12,DVR77
```

Dummy driver

```
/E
```

```
* DEVICE REFERENCE TABLE

 1 = EQT #?
2

 2 = EQT #?
1,1

 3 = EQT #?
0,0

 4 = EQT #?
5

 5 = EQT #?
4

 6 = EQT #?
6,1

 7 = EQT #?
7,0

 8 = EQT #?
7,1

 9 = EQT #?
0

10 = EQT #?
3

11 = EQT #?
16

12 = EQT #?
8

13 = EQT #?
9

14 = EQT #?
10

15 = EQT #?
11

16 = EQT #?
12

17 = EQT #?
13
```

Logical Unit numbers 1 through 6 are predefined in the RTE System as:

1 - System teleprinter
2 - System mass storage
3 - Auxiliary mass storage
4 - Standard punch unit
5 - Standard input unit
6 - Standard list unit

```
18 = EQT #?
14

19 = EQT #?
15

20 = EQT #?
1,6

21 = EQT #?
1,2

22 = EQT #?
1,3

23 = EQT #?
1,4

24 = EQT #?
1,5

25 = EQT #?
1,6

26 = EQT #?
1,7

27 = EQT #?
0

28 = EQT #?
0

29 = EQT #?
0

30 = EQT #?
/E

* INTERRUPT TABLE

13,EQT,2

15,EQT,4

16,EQT,5

17,EQT,6

20,EQT,7

21,EQT,1
```

Interrupt table generation requires an entry for each I/O card in ascending order.

```
22,EQT,1

23,EQT,6

24,EQT,5

25,PRG,QUEUE

26,PRG,QUEUE

27,PRG,QUEUE

30,PRG,QUEUE

31,PRG,QUEUE

32,PRG,QUEUE

33,PRG,QUEUE

34,PRG,QUEUE

/E
```

Interrupt table complete.

```
LIBRARY

   QUIES    32172
   .ENTR    32574
   PRTN     32764

EP LINKAGE 32527
```

RTGEN outputs the names and entry point addresses of all library routines which are referenced by Real-Time and background programs.

New first word of available core memory in base page.

```
RT RESIDENTS

QUEUE(20)   31767

DISP (14)   31710

TAM  (35)   31741

ERR  (36)   32101
   RMPAR    32222
   GETAD    32245

D.RTP(41)   32263
   P.PAS    34155

RP LINKAGE  32526

RT DISC RESIDENTS
```

RTGEN begins loading the Real-Time core-resident programs.

RTGEN begins loading the Real-Time disc-resident programs.

```
RE&M (3%)    342%4
  R/W%       42115
  RMPAR      42174
  GETAD      42217

DISC (3%)    342%4
  RMPAR      36%73
  NAMF       36116
  FCONT      36276
  LOCF       36362
  APOSN      36542
  CREAT      36651
  PURGE      37140
  OPEN       37237
  READF      37422
  POSNT      4%2%7
  RWNDF      4%444
  CLOSE      4%526
  FSTAT      4%635
  GETAD      4%662
  NAM..      4%7%%
  P.PAS      40775
  RWSUB      41%24
  $OPEN      41272
  R/W%       41447
  RWNDF      41526

BP LINKAGE %%743

CHANGE OF LINKAGE?
11%%

FWA SY MEM 42235

CHANGE FWA SYS AV MEM?
45%%%

BG BOUNDRY?
47%%%

BG RESIDENTS

(NONE)

BG DISC RESIDENTS

PROGL(2%)    47%%%
  .EAU.      5%143
  CLRIO      5%213
  IAND       5%216
  RMPAR      5%226
  OPEN       5%251
  READF      5%%34
```

Report of new first word of available core memory in base page.

Linkage area is changed for future addition of larger Real-Time disc-resident programs on-line.

First word address of system available memory.

Address change is to increase the Real-Time disc resident area for future addition of larger programs.

First address of the background area (area from 47000_8 to 50000_8 is used for temporary storage of output buffers, and reentrant temporaries).

Begin load of background disc-resident programs, with names and entry points for mains and subroutines.

```
   DLKDS     51221
   CLOSE     51273
   GETAC     51402
   $OPEN     51420
   R/L.      51575
   P.PAS     51654
   RN$UB     51703
   RWND9     52151

DLIST(31)    47000
   FSTAT     52260

ASMB (99)    47000

ASMBD(99)    53643

ASMB1(99)    53643

ASMB2(99)    53643

ASMB3(99)    53643

ASMB4(99)    53643

ASMB5(99)    53643

ALGOL(99)    47000
   .EAU.     51312
   %WRIT     51362
   SREAD     52063
   .OPSY     52621

ALGL1(99)    52661

FTN4 (99)    47000

FTN40(99)    55311

FTN41(99)    55311

FTN42(99)    55311

FTN43(99)    55311

FTN44(99)    55311

FTN45(99)    55311

FTN46(99)    55311

FTN47(99)    55311

FTN48(99)    55311
```

```
FTN49(99)   55311

FTN4A(99)   55311

FTN4B(99)   55311

FTN4C(99)   55311

FTN4D(99)   55311

FTN4E(99)   55311

FTN4F(99)   55311

FTN4G(99)   55311

FTN4H(99)   55311

EDIT (99)   47000
  SREAD     51411
  %RIS      52147
  .OPSY     52437

LOADR(99)   47000

SXL  (99)   47000
  SGMTR     47047
  SPROC     47134
  LDUMM     47325
  DBUGS     50367
  .EAU.     51305
  SPTRP     51555
  DTSTR     52434
  PREUP     52577
  FD5S      53251
  JSBES     53317
  ULIBR     53324
  ALLOC     53372
  PUT4      53560
  DIAG      53630
  PAGER     53654
  .DRCT     54737

SG01L(99)   54746
  SCNSM     55056
  SCNP1     55761
  LCRST     56160
  SCNMM     56633
  FD5T      57024
  FD5V      57057
  NXSY      57123
  SNAM      57210
```

```
   MAT1        57437
   MAXR        57500
   PARSE       57652
   SCGRM       60125

SG02L(9F)      54746
   SNEST       55056
   LDUM2       55255
   SNTX        55461
   UGRMF       55737
   FKDCR       56727
   FUST        57011
   FUSV        57044
   FSEMI       57110
   PARSE       57245
   FUO         57520

SG03L(9F)      54746
   ENDR        55056
   SWTCH       56047
   BLNK        56110
   MOVE.       56126
   POSN.       56145
   RVERF       56212
   OCTAL       56456
   MAXR        56546
   POLG        56620
   REDGL       57015
   .LOCF       57153
   READF       57177
   R/WR        57764
   .XEC        60043
   RODSK       60113
   P.PAS       60241
   RNSUB       60270
   RNPOS       60536

SG04L(9F)      54746
   ENDR2       55056
   .XEC        55476
   MSTBL       55546
   PRINT       55650
   CFXUF       56012
   BLOK        56413
   CCON        56547
   STFN        56572
   SNAM        56617
   FUST        57046
   FUSV        57101
   FUSA        57145
   BLNK        57210
   OCTAL       57226
   MAXR        57316
```

```
   PRPLK     57370
   ABOUT     57444
   NXSY      57545
   STMA      57632
   READF     60022
   R/WS      60607
   P.PAS     60666
   RWSUB     60715
   RWNDS     61163

SG05L(S-V)   54746
   LDUMA     55456
   FELST     55346
   TERMF     55642
   FUSV      55670
   MSTBL     56734
   TRBAK     57036
   BLNK      57201
   OCTAD     57217
   CLOSE     57307
   NXSY      57416
   FUST      57503
   OPEN      57536
   APOSN     57721
   FKDCH     60044
   MOVE.     60122
   R/WS      60141
   RMPAR     60220
   $OPEN     60243
   RWSUB     60420
   LOCF      60666
   GETAD     61046
   RWNDS     61064
   P.PAS     61174

SG06L(S-V)   54746
   DGLST     55056
   FUSV      56515
   NXSY      56551
   CCON      56646
   MSTBL     56671
   .XEC      56773
   .GOTO     57043
   UNSTR     57066
   OCTAD     57170
   FACTR     57264
   MOVE.     57434
   LOCF      57453
   CLOSF     57633
   FKDCH     57742
   DIVD      60024
   P.PAS     60042
   R/WS      60071
```

```
SG97L(94)   54746
  LOUM4     55V56
  XFRST     55346
  FU5V      55613
  MSTBL     55657
  FU5T      55761
  TWBAK     56014
  OPEN      56157
  ILOSE     56342
  LOCF      56365
  PROCB     56545
  MOVE.     56627
  BLNK      56646
  APOSN     56664
  CLOSF     57003
  RMPAM     57112
  %OPEN     57135
  P.PAS     57312
  RW%UB     57341
  R/W%      57407
  GETAD     57466
  RWNDS     57704

SG98L(94)   54746
  DSPS1     55056
  DSUDF     55436
  DCTAL     55652
  MSTBL     55742
  BLNK      56044
  FU5V      56062
  FU5T      56126
  NXSY      56151
  CFXUP     56246
  UNSTR     56647
  CCON      56751
  MAXR      56774
  PUPLK     57046
  ABOUT     57122
  READF     57223
  R/W%      57410
  P.PAS     57467
  RW%UB     57516
  RWNDS     57564

SG99L(94)   54746
  SETSM     55056
  FU5T      55000
  FU5V      55633
  STMA      55677
  ABOUT     56067
  PRFJT     56170
  CFXUP     56332
  BLNK      56735
```

```
   READF      56751
   MAXR       57536
   MSTBL      57610
   OCTAD      57712
   PRPLR      60002
   R/WS       60056
   P.PAS      60135
   RWSUR      60164
   RWNDS      60432

SG1WL(99)     54746
   OTPT       55056
   READF      56632
   OPEN       57417
   CLOSE      57602
   MSTBL      57711
   ABOUT      60013
   MOVE.      60114
   BLNK       60133
   FDSV       60151
   R/WS       60215
   P.PAS      60274
   RWSUR      60323
   RMPAR      60571
   SOPEN      60614
   MAXR       61771
   RWNDS      61043
   GETAD      61153

SG11L(99)     54746
   NAMR       55056
   SCOMM      55635
   .LOCF      55736
   FDSV       55762
   GWNS       56026
   MAXR       56077
   STMA       56151
   SWTCH      56341
   RVEMF      56402
   MOVE.      56646
   GET4       56765
   POLG       56721
   REDGL      57116
   POSN.      57254
   READF      57321
   .XEC       60106
   RDDSK      60156
   R/WS       60304
   P.PAS      60363
   RWSUR      60412
   RWNDS      61056
```

```
SG12L (9₹)    54746
  I001        55₹56
  SKIPF       55355
  FU5T        55544
  STFM        55577
  GNMS        55624
  STMA        55675
  MAT1        56₹65
  RVERF       56226
  SWTCH       56472
  MOVE.       56533
  MXSY        56552
  GET4        56637
  .XEC        56673
  POLG        57743
  REDGI       57140
  POSW.       57276
  .LOCF       57343
  HEADF       57367
  RDGSP       57154
  R/W$        57302
  F.PAS       57361
  RWSUB       57417
  RWND$       57656

SG13L (9₹)    54746
  OTPST       55₹56
  LDM12       55416
  FU5T        55671
  FU5V        55724
  FKDCP       55770
  CREAT       56₹52
  MSTBL       56331
  OPEN        56433
  BLNK        56616
  CLOSE       56634
  SOPEN       56743
  NAM..       57120
  RMPAR       57215
  R/W$        57247
  PWNL$       57317
  GETAD       57427
  RWSUB       57445
  P.PAS       57713

SG14L (9₹)    54746
  INIT2       55₹56
  AEND        56137
  CRDMC       55672
  DSUDF       55541
  MSTBL       55755
  CREAT       56₹57
  .XEC        56336
```

```
   MOVE.     56406
   CFXUP     56425
   UNSTR     57026
   BLNK      57130
   CCON      57146
   CLOSF     57171
   $OPEN     57300
   NAM..     57455
   RMPAR     57552
   MAXR      57575
   PBPLK     57647
   ABOUT     57723
   R/W$      60024
   RWNDX     60103
   GETAL     60213
   READF     60231
   RWSUB     61016
   P.PAS     61264

SG15L(99)    54746
   CLOP      55056
   PKUDF     55073
   CLOSF     55157
   R/W$      55266

SG16L(99)    54746
   DBL       55056
   ABOUT     56061
   SWTCH     56162
   RVERF     56223
   FU5V      56467
   MAXR      56533
   LLINK     56605
   READF     57215
   POLG      60002
   REDGL     60177
   PUSH.     60335
   .LOCF     60442
   .XEC      60426
   PBPLK     60476
   R/W$      60552
   P.PAS     60631
   RWSUB     60661
   RDDSK     61126
   RWNDX     61254

SG17L(99)    54746
   RDCRD     55056
   READF     55304
   TRBAK     56071
   R/W$      56234
   P.PAS     56513
   RWSUB     56542
```

```
   OPEN      56610
   APOSN     56773
   FKDCF     57112
   MOVE.     57174
   RWNDT     57213
   CLOSF     57323
   RMPAR     57432
   $OPEN     57455
   LOCF      57532
   GETAD     60012
SG18L(99)    54746
   CRTOP     55056
   FU5V      55201
   BLNK      55245
   MSTBL     55263
   CREAT     55365
   CLOSE     55644
   $OPEN     55753
   NAM..     56130
   RMPAR     56225
   R/W5      56250
   RWNDS     56327
   GETAD     56437
   PW$UP     56455
   P.PAS     56723

SG19L(99)    54746
   SNPOP     55056
   CCON      56367
   NXSY      56412
   FU5V      56477
   FU5T      56543
   UNSTF     56576
   READF     56700
   MSTBL     57065
   OCTAD     57567
   ABOUT     57657
   BLNK      57760
   R/W5      57776
   P.PAS     60055
   RW$UR     60104
   MAXR      60352
   RWNDS     60424

SG20L(99)    54746
   LNKST     55056
   MAPST     55137
   SGINT     55501
   FU5V      55671
   DINIT     55735
   INIT1     56213
   .XEC      56417
```

```
   MAXR        56467
   MOVE.       56541
   SYM         56569
   MEMR        57361
   OPEN        57422
   FKDCB       57605
   CLOSE       57667
   RMPAR       57776
   $OPEN       60021
   R/W$        60176
   GETAD       60255
   R%NDS       60273
   R%SUP       60403
   P.PAS       60651

SG21L(99)      54746
   OPLG        55056
   SNAM        55637
   FU5T        56066
   FU5V        56121
   FU5A        56165
   CFXUP       56230
   PRENT       56631
   MAT2        56773
   CCON        57115
   BLNK        57140
   MSTBL       57156
   MAXR        57260
   PBPLK       57332
   ABORT       57406
   OCTAO       57507
   READF       57577
   R/W$        60364
   P.PAS       60443
   R%SUP       60472
   R%NDS       60740

FMGR (99)      47000
   FM.CM       47643
   .EAU.       51451
   .DRCT       51521
   READF       51530
   CLOSE       52315
   OPEN        52424
   R/W$        52607
   P.PAS       52566
   RMSUP       52715
   RMPAR       53163
   $OPEN       53206
   R%NDS       53363
   GETAD       53473
```

```
FMGR0(99)   53511
  PK..      53517
  CR..      55145
  CN..      55251
  FM.UT     56313
  LOCK.     57436
  RWNDF     57506
  ID.A      57570
  CREA.     57710
  NAM..     57762
  NAMF      60457
  CREAT     60231

FMGR1(99)   53511
  .PARS     53537
  REA.C     54453
  C.TAB     54536
  EE..      54617
  TR..      54654
  SA..      55045
  MR..      55307
  CREA.     55146
  LOCF      56220
  READ.     56400
  RWNDF     56425
  CK.SP     56507
  WRLG      56623
  CREAT     56641
  SREAD     57120
  %ARIT     57656
  NAM..     60357
  .OPSY     60454

FMGR2(99)   53511
  IN.IT     53520
  IN..      54372
  RC..      56210
  RC..      56520
  FM.LT     56707
  IPUT      60032
  FID.      60053
  NAM..     60173
  J.PUT     60270
  LOCK.     60315
  MSG.      60365

FMGR3(99)   53511
  LI..      53517
  DL..      55230
  PU..      56453
  FM.UT     56676
  LOCF      60021
  MSG.      60201
```

```
   LOCK.      52236
   PURGE      52396

FMGR4(99)     53511
   ST.UP      53520
   CO..       54724
   SP..       55431
   CREA.      56206
   LOCF       56260
   CK.SP      56440
   FM.UT      56554
   CREAT      57677
   PWNDF      60156
   ID.A       60240
   NAM..      60360

FMGR5(99)     53511
   ??..       53523
   MS..       55564
   PP..       56063
   CL..       56705
   F.UTM      57167
   WRISS      57537
   IPUT       57576
   ID.A       57417
   LOCF       57537
   FSTAT      57717
   %WRIS      57744

LSTEN(99)     47700
   CREAT      51534
   CLOSE      52013
   RMPAR      52122
   PURGE      52145
   $OPEN      52244
   NAM..      52421
   R/W$       52516
   GETAD      52575
   OPEN       52613
   R%ND$      52776
   RW%UB      53106
   P.PAS      53354

BP LINKAGE   1633

SYSTEM STORED ON DISC
SYS SIZE: 25 TRKS, 400 SECS(1A)
```

SECTION VII

SAMPLE PROGRAMS

The sample programs in this section are provided to illustrate coding sequences for various HP 91701 system concepts discussed in this manual. Examples 1 and 2 illustrate program-to-program communication between a program at the central computer and a program at a terminal. Examples 3 and 4 illustrate the master/slave concept which exists in the HP 91701 software. The terminal computer provides the data which the central computer processes. Example 4 is provided to illustrate some of the material described in Appendix A.

EXAMPLE I

Terminal Receives Data

This example shows the code which can be used at the terminal and at central to produce program-to-program communication. The data transmission in this example is from the central computer to the terminal. The data block transfer must be initiated at central by a "transmit request" which must then be received by the terminal. Code is included in this example to permit the terminal operator to abort the program at the terminal and terminate central. The actual program-to-program communication is accomplished via #TAM calls instead of calls directly to D.65 via .IOC.

```
******* TERMINAL PROGRAM ******* *
                                *
     EXT PTPON,PTPOF,ESCON,IDLE *
     EXT GETLU,REXEC,#TAM       *
     .                          *
     .                          *
     .                          *
     JSB PTPON  DISABLE OPERATOR*
     DEF *+1    ATTENTION.      *
                                *
     JSB GETLU  GET LOGICAL UNIT*
     DEF *+2    AT CENTRAL FOR  *
     DEF LU     THIS TERMINAL.  *
                                *
SCHD JSB REXEC  SCHEDULE ABCD   *
     DEF *+5    AT CENTRAL.     *
     DEF STATS                  *
     DEF B12                    *
     DEF PGNAM  ASCII "ABCD  ". *
     DEF LU     PASSED PARAMETER*
                                *
     SZA        ABCD DORMANT?   *
     JMP SCHD   NO, RE-SCHEDULE.*
                                *
     CLA,INA    READ REQUEST    *
     JSB #TAM   FROM CENTRAL.   *
     DEF *+5                    *
     DEC 5                      *
     DEF BUFFR                  *
     DEC -2     1 WORD REQUEST. *
     DEF *+1    COMPLETOR SUBR. *
                                *
     NOP                        *
     JSB ESCON  CHECK IF OPERATOR*
     DEF *+2    HAS PRESSED     *
     DEF TEMP   ESCAPE KEY.     *
     LDA TEMP                   *
     SZA                        *
     JMP XSTP   YES, SEND STOP. *
                                *
     CLA        READ DATA FROM  *
     JSB #TAM   CENTRAL.        *
     DEF *+5                    *
```

```
***** CENTRAL PROGRAM ABCD *****

          .
          .
          .
          LDA 1,I    GET PASSED LU.
          STA LU

          JSB EXEC   SEND REQUEST
          DEF *+6    AND DATA.
          DEF B2     REQ. CODE = WRITE.
          DEF LU     LU OF TERMINAL.
          DEF PBUFR  PARAM BUFFER.
          DEF B4     PARAM BUFFER LENGTH.
          DEF B10    TRANSMISSION MODE.
                .
          SLA,RSS    ANY ERRORS?
          JMP ERROR  YES.
          .          NO, CONTINUE
          .          PROCESSING.
          .
*ERROR    ALF,ALF    WAS STOP COMMAND
          ALF,SLA    RECEIVED?
          JMP DONE   YES, TERMINATE.
          .          NO, CHECK FOR
          .          OTHER ERRORS.
          .
```

```
      OCT 5                          *
      DEF BUFFR                      *
      DEC -40     20 WORD RECORD.    *
      DEF *+1     COMPLETOR SUBR.    *
                                     *
      .           READ & PROCESS     *       .
      .           TILL DONE.         *       .
                                     *       .
DONE  JSB PTPOF   ENABLE OPERATOR    *DONE   JSB EXEC    TERMINATE ABCD.
      DEF *+1     ATTENTION.         *       DEF *+2
      JSB IDLE    RETURN TO TCE/3.   *       DEF B6
                                     *
XSTP  JSB #TAM    TRANSMIT STOP      *PBUFR  DEF DBUFR   DATA BUFFER.
      DEF *+2       TO ABCD.         *       DEC 20       (20 WORDS)
      DEC 2                          *       DEF RBUFR   REQUEST BUFFER.
      JMP DONE    EXIT.              *       DEC 1        (1 WORD)
```

EXAMPLE II

Terminal Sends Data

This example shows code which can be used to produce program-to-program communication initiated by the terminal to send data to the central computer. The data block transfer must be initiated at central by a "transmit request," which must then be received by the terminal. The data transfer is then made from the terminal to central (in contrast to Example 1). Transmission across the data link is also accomplished via #TAM calls rather than D.65.

```
******* TERMINAL PROGRAM ******* *
                                *
     EXT PTPON,PTPOF,ESCON,IDLE *
     EXT GETLU,REXEC,#TAM       *
     .                          *
     .                          *
     .                          *
     JSB PTPON  DISABLE OPERATOR*
     DEF *+1    ATTENTION.      *
                                *
     JSB GETLU  GET LOGICAL UNIT*
     DEF *+2    AT CENTRAL FOR  *
     DEF LU     THIS TERMINAL.  *
                                *
SCHD JSB REXEC  SCHEDULE ABCD   *
     DEF *+5    AT CENTRAL.     *
     DEF STATS                  *
     DEF B12                    *
     DEF PGNAM  ASCII "ABCD  ". *
     DEF LU     PASSED PARAMETER*
                                *
     SZA        ABCD DORMANT?   *
     JMP SCHD   NO, RE-SCHEDULE.*
                                *
     CLA,INA    READ REQUEST    *
     JSB #TAM   FROM CENTRAL.   *
     DEF *+5                    *
     DEC 5                      *
     DEF BUFFR                  *
     DEC -2     1 WORD REQUEST. *
     DEF *+1    COMPLETOR SUBR. *
                                *
     NOP                        *
     JSB ESCON  CHECK IF OPERATOR*
     DEF *+2    HAS PRESSED     *
     DEF TEMP   ESCAPE KEY.     *
     LDA TEMP                   *
     SZA                        *
     JMP XSTP   YES, SEND STOP. *
                                *
     CLA        SEND DATA TO    *
     JSB #TAM   CENTRAL.        *
     DEF *+4                    *
     OCT 6                      *
     DEF BUFFR                  *
     DEC -40    20 WORD RECORD. *
```

```
***** CENTRAL PROGRAM ABCD *****
     .
     .
     .
     LDA 1,I    GET PASSED LU.
     STA LU

     JSB EXEC   SEND REQUEST
     DEF *+6    AND RECEIVE DATA.
     DEF B1     REQ. CODE = READ.
     DEF LU     LU OF TERMINAL.
     DEF PBUFR  PARAM BUFFER.
     DEF B4     PARAM BUFFER LENGTH.
     DEF B0     TRANSMISSION MODE.

     SLA,RSS    ANY ERRORS?
     JMP ERROR  YES.
     .          NO, CONTINUE
     .          PROCESSING.
     .
*ERROR ALF,ALF  WAS STOP COMMAND
     ALF,SLA    RECEIVED?
     JMP DONE   YES, TERMINATE.
     .          NO, CHECK FOR
     .          OTHER ERRORS.
     .
```

```
      .           READ & PROCESS
      .           TILL DONE.

DONE  JSB  PTPOF  ENABLE OPERATOR
      DEF  *+1    ATTENTION.
      JSB  IDLE   RETURN TO TCE/3.

XSTP  JSB  #TAM   TRANSMIT STOP
      DEF  *+2      TO ABCD.
      DEC  2
      JMP  DONE   EXIT.
```

```
*
*
*       .
*       .
*       .
*DONE   JSB  EXEC   TERMINATE ABCD.
*       DEF  *+2
*       DEF  B6
*
*PBUFR  DEF  DBUFR  DATA BUFFER.
*       DEC  20      (20 WORDS)
*       DEF  RBUFR  REQUEST BUFFER.
*       DEC  1       (1 WORD)
```

EXAMPLE III

Enable Polling Mode

The code sequence shown in this example is used by a terminal program to receive a request via the Polling mode (rather than the Listen mode). The .IOC. calls to D.65 are described in Appendix A. The Interrupt Special Open Loop is included here to illustrate the communication between a DMA central and a non-DMA terminal described in Appendix A.

```
        JSB .IOC.       ENABLE POLLING MODE
        OCT 303YY
        JMP *-2

RECV    JSB .IOC.       RECEIVE REQUEST
        OCT 101YY
        JMP *-2
        DEF BUFF
        DEC 3

        JSB .IOC.       STATUS REQUEST
        OCT 400YY
        SSA             COMPLETE? (A)=EOT STATUS WORD
        JMP *-3         NO,KEEP WAITING

        ALF,ALF         YES;WAS REQUEST RECEIVED?
        RAL              (IF NOT,BIT 6 IS SET)
        SSA
        JMP RECV        NO,REPEAT READ CALL

        ALF,ALF         YES,CHECK FOR ERRORS
        RAR               (IF NONE,BIT 0 IS SET)
        SLA
        JMP ERROR       ERROR:GO CHECK

        <continue>      NO ERROR,CONTINUE.
         .
         .
BUFF    OCT 42          INTERRUPT SPECIAL OPEN LOOP
        DEF IRBUF
        DEC IRBFL
IRBUF   BSS 5           5-WORD REQUEST BUFFER
IRBFL   DEC 5
```

EXAMPLE IV

Central Processes Data

This example is provided to show a FORTRAN coding sequence and to illustrate terminal collection and central processing of data. The terminal program accepts up to 128 numbers, converts them to binary and sends one record across to the central computer. The terminal program schedules a central program (DEMOC) to read the file and do the processing.

```
      PAGE  0001              (FTN4--RELEASE 24177B--JULY, 1971)
0001  C
0002  C
0003  C
0004  C
0005  C               TERMINAL PROGRAM
0006  C
0007  C
0008  C
0009  C
0010  FTN4,L
0011        PROGRAM NUMB
0012  C     PROGRAM  TO READ IN NUMBER FROM REMOTE TERMINAL
0013        DIMENSION IARRY(128),INAM(3),ISIZE(2),IPNAM(3),IREG(2)
0014        INTEGER OUT,OK
0015        EQUIVALENCE (REG,IREG),(IA,IREG(1)),(IB,IREG(2))
0016        INAM(1)=2HFI
0017        INAM(2)=2HLE
0018        INAM(3)=2H##
0019        OUT=2
0020        IN=1
0021        IEND=9999
0022        OK=-1
0023        WRITE(OUT,10)
0024   10   FORMAT ("PROGRAM TO ACCEPT NUMBERS FROM A REMOTE TERMINAL"/
0025       1"AND STORE THE NUMBERS ON A FILE AT THE CENTRAL ."/
0026       2"INPUT NUMBERS,TERMINATE ENTRIES WITH 9999.  MAX OF 128 NUMBERS"/
0027        ICNT=1
0028   100  ICNT=ICNT+1
0029        READ (IN,*) INUMB
0030        WRITE (OUT,20)INUMB
0031   20   FORMAT (I4)
0032        IARRY(ICNT)=INUMB
0033        IF (INUMB-IEND) 105,110,105
0034   105  IF (ICNT-128) 100,110,110
0035   110  ICNT=ICNT-1
0036        IARRY(1)=ICNT
0037        CALL ROPEN(ISTAT,IERR,INAM)
0038        IF((ISTAT.EQ.OK).AND.(IERR.GE.0)) GO TO 120
0039        WRITE (OUT,30)
0040   30   FORMAT("OPEN ERROR, PROGRAM ABORTED")
0041        GO TO 9999
0042   120  CALL RWRIT(ISTAT,IERR,INAM,IARRY,ICNT)
0043        IF ((ISTAT.EQ.OK).AND.(IERR.GE.0)) GO TO 130
0044        WRITE(OUT,40)
```

```
0045    40    FORMAT("WRITE ERROR, PROGRAM ABORTING")
0046          GO TO 9990
0047   130    CONTINUE
0048          WRITE(OUT,60)
0049    60    FORMAT("DO YOU WANT TO SCHEDULE CENTRAL PROGRAM=YES ON NO?")
0050          READ (IN,70) IARRY(1)
0051    70    FORMAT(A2)
0052          IF (IARRY (1).NE.2HYE) GO TO 9990
0053          CALL RCLOS(ISTAT,IERR,INAM)
        PAGE  0002  NUMB       (FTN4--RELEASE 24177B--JULY, 1971)
0054          IF((ISTAT.EQ.OK).AND.(IERR.GE.0)) GO TO 140
0055          WRITE(OUT,50)
0056    50    FORMAT("CLOSE ERROR, PROGRAM ABORTING")
0057          GO TO 9990
0058   140    CONTINUE



0059          ICODE=10
0060          WRITE (OUT,95)
0061    95    FORMAT ("INPUT NAME OF PROGRAM TO SCHEDULED"  )
0062   C      READ PROGRAM NAME FROM TTY
0063          READ (IN,96)IPNAM
0064    96    FORMAT (3A2)
0065          REG= REXEC(ISTAT,ICODE,IPNAM)
0066          IF (ISTAT.EQ.OK)  GO TO 150
0067          WRITE(OUT,80)ISTAT
0068    80    FORMAT("SCHEDULE COMM LINE ERROR",I6,"ABORTING")
0069          GO TO 9990
0070    150   CONTINUE
0071          IF (IA.EQ.0) GO TO 9990
0072          WRITE (OUT,90)
0073    90    FORMAT ("PROGRAM BUSY...ABORTING")
0074    9990  CALL IDLE
0075          END
**  NO ERRORS*
        PAGE  0003  NUMB       (FTN4--RELEASE 24177B--JULY, 1971)
0076        ENDS


        PAGE  0001             (FTN4--RELEASE 24177B--JULY, 1971)
0001   C
0002   C
0003   C
0004   C
0005   C         CENTRAL PROGRAM
0006   C
0007   C
0008   C
0009   C
0010   C
0011   FTN4,L
```

```
0012          PROGRAM DEMOC
0013    C     PROGRAM TO READ FROM A FILE CALLED FILE## AND PRINT THE NUMER
0014    C     SQUARE OF THE NUMBER, AND THE SQUARE ROOT OF THE NUMBER.
0015          DIMENSION IDCB(144),INAM(3),IBUF(128)
0016          INTEGER CRT,LPT
0017          CRT=6
0018          LPT=6
0019          INAM(1)=2HFI
0020          INAM(2)=2HLE
0021          INAM(3)=2H##
0022          WRITE (LPT,10)
0023    10    FORMAT("1NUMBER     SQUARE      SQUARE ROOT     CUBE      CUBE ROOT"/
0024          CALL OPEN (IDCB,IERR,INAM)
0025          IF(IERR.GE.0) GO TO 100
0026          WRITE (CRT,20)
0027    20    FORMAT(" OPEN ERROR, ABORTING")
0028          GO TO 9990
0029    100   CALL READF (IDCB,IERR,IBUF)
0030          IEOF=-12
0031          IF (IERR-IEOF) 105,9900,105
0032    105   CONTINUE
0033          IF (IERR.GE.0) GO TO 110
0034          WRITE (CRT,30)IERR
0035    30    FORMAT (" READ ERROR, ABORTING...ERROR WORD=",I8)
0036          GO TO 9900
0037    110   DO 120 I=2,IBUF(1)
0038          INUM=IBUF(I)
0039          ANUM=INUM
0040          ANUMS=ANUM*ANUM
0041          ANUMQ=ANUM*ANUMS
0042          ANSR=0
0043          ANQR=0
0044          IF (INUM)  130,130,140
0045    140     CONTINUE
0046          ANSR=ANUM**(1./2.)
0047          ANQR=ANUM**(1./3.)
0048    130     CONTINUE
0049          WRITE (LPT,40) INUM,ANUMS,ANSR,ANUMQ,ANQR
0050    40    FORMAT (1X,I6,3X,F8.0,4X,F11.4,2X,F8.0,3X,F9.4/)
0051    120   CONTINUE
0052          GO TO 100
0053    9900  WRITE(CRT,50)
       PAGE  0002  DEMOC      (FTN4--RELEASE 241778--JULY, 1971)
0054    50    FORMAT(" END OF PROGRAM"//////)
0055    9990  CALL CLOSE(IDCB)
0056          END
**  NO ERRORS*
       PAGE  0003  DEMOC      (FTN4--RELEASE 241778--JULY, 1971)



1057          ENDS
```

APPENDIX A

COMMUNICATION DRIVER INTERFACE

This appendix has been taken from the Computer Serial Interface Kit Manual HP 12771A, Section III. It is reproduced here as an aid to the user in the construction of communication driver interfaces.

APPENDIX A

COMMUNICATION DRIVER INTERFACE

INTRODUCTION

Two general-purpose software drivers are furnished which were designed to satisfy the requirements of a wide variety of applications. This section describes driver characteristics, call sequences to the drivers, data transmission mode compatibility, and programming techniques.

DRIVER CHARACTERISTICS

The task of the drivers is to accurately transfer a word from storage in one computer system to storage in another computer system. To effect such transfers, the driver for one computer system must communicate with the driver in the other computer system, effectively using the hardware interface between computers. Since one computer may be BCS configured and the other may operate in the RTE environment, BCS and RTE drivers communicate in the same way.

CLOSED LOOP OPERATION

In closed loop operations, both cards are in the receive operating mode. The RTE driver does not use this mode for data transmissions. Both drivers use this mode for request transmissions. The transmitting computer, using one of the drivers and user-written software, transmits two driver parameters followed by the user's request parameters.

Driver parameters are two words generated by the transmitting driver for interpretation by the receiving driver. These words precede every request transmission. One word advises the request buffer length of the request transmission to follow; the second reports the data transmission mode and if data is to follow. The receiving driver checks the buffer length and data transmission mode compatibility. Data transmission mode compatibility is described later in this section.

The requests make possible the two-way communication between the user-written software in one computer and the user-written software in another computer. Request parameters provide information such as the name of the buffer to be transmitted or that is requested, the desired transmission mode (when more than one is compatible), and the length of the data block. The drivers place no restrictions on the number, format, or information content of request parameters. A request must always precede the transmission of a data block. The request may be originated by the transmitting software and immediately followed by data or by the receiving software which requested data.

The word "data" used in this section implies the information, other than requests, to be transferred from one computer to another. Data may be one word or a block of words. The data may be a user-written test program, data linearization or modification factors, data from a data acquisition system, etc.

For each word transmitted in closed loop, the receiving card checks parity. The driver software checks parity by reading the card status word. If parity is not correct, a reply to the transmitting software directs retransmission of the word. If a parity error has not occurred, the card directs transmission of the next word or ends transmission at the end of the block. The transmitting software will always wait for a reply.

OPEN LOOP OPERATION

In open loop operation, the transmitting card is in the transmit operating mode and the receiving card is in the receive operating mode. Open loop is used only for data transmissions. The words which comprise the data are transmitted with approximately ten microseconds between words. The receiving software does not reply to the words received; the transmitting software does not wait for a reply. The receiving card, however, checks parity. After receipt of the last word, the transmitting card is set to the receive operating mode and the receiving software checks parity. If any one parity error occurred during the entire open loop transmission, the transmitting software is directed to return to open loop operation and to retransmit the entire block of data words.

DMA SPECIAL OPEN LOOP OPERATION

In DMA special open loop operation, both cards are in the receive operating mode. Special open loop is used only when the transmitting computer has DMA and the receiving computer does not. The transmitting software, after transmission of a word, waits for a reply before transmitting the next word. The receiving software always directs that the next word be transmitted and never requests retransmission of the previous word. This allows a computer having the DMA capability to have transmission communication with a non-DMA (interrupt) computer. After receipt of the last word, the receiving software checks parity. If any parity error occurred during the entire open loop transmission, the transmitting software is directed to return to DMA special open loop operation and to retransmit the entire block of data words.

DATA TRANSMISSION MODES

The choice of data transmission mode is determined by system hardware (DMA or no DMA) and system operating requirements. Compatible modes are shown in Table A-7. If DMA is available, the most efficient mode is DMA open loop write to DMA open loop read. DMA is always present in the RTE system and is always used. DMA may or may not be present in a BCS system.

When transferring between a computer with DMA and a computer without DMA, the most efficient mode is DMA to dedicated open loop. This will provide the same data transfer rate as DMA to DMA at the cost of turning off the interrupt system during data transfer.

When transferring between DMA and non-DMA computers where the program cannot allow the interrupt system to be turned off in the non-DMA computer, two modes are available. They are Interrupt open loop write to DMA open loop read and DMA special open loop write to Interrupt Special open loop read.

When transferring between two non-DMA computers, the primary decision will be whether to turn the interrupt off and use a dedicated mode (provides same speed as DMA) or to leave the interrupt on in one or both computers and use the slower interrupt mode.

BCS DRIVER D.65

Open and closed loop transmissions are accomplished using interrupt, dedicated (SFS), or DMA transfers. The request receiving driver checks data transmission mode compatibility between transmitter and receiver as described later in this section. The Assembler calling sequences to the driver are summarized in Table A-1. To make a FORTRAN call possible, a subroutine named L65 is provided to set up the proper IOC calling sequences. The FORTRAN calls are summarized in Table A-2.

TRANSMIT REQUEST AND RECEIVE OR TRANSMIT DATA

This call, item 1 of Tables A-1 and A-2, must be used to pass data. The second computer will read the request with a Receive Request Only call, and after examination of the request parameters, will take appropriate action.

If one computer makes a transmit request and transmit data call, and another computer after reading the request does not wish to (or cannot) transfer the data at this time, it must reply with a STOP before other calls will be allowed by the driver. This problem is most likely to occur when polling, where the polling computer reads the request but wishes to poll another device before handling a data transfer requested by the first device. If this is attempted, the the program will hangup. When the first request includes a data transfer, the transfer must be accomplished or rejected before other calls to the driver are allowed.

If, when reading the request, the driver in the second computer finds that the Receive Request call specified no or too small of a request buffer, the request will be rejected. Status in both computers would be Request Not Accepted.

After the request is read, the user program must now decide its course of action. The user has the option of declining the data transfer (too large, didn't specify size, program doesn't exist) by sending a STOP reply (Status in # 1 would show Data Transmission not initiated) or accepting with a Receive Data call.

If the data transfer request is to be accepted, the user makes a Receive or Transmit Data call. At this time the driver in the request receiving computer checks the data transmission mode information received from the initiating computer against the mode information contained in the Receive or Transmit Data call. If it finds the modes incompatible, it will send a STOP reply to the initiating computer and return to the user's program with status showing Incompatible Mode. The driver in the initiating computer will return with Data Transfer Not Initiated bit set.

NOTE

The data buffer length must be exactly the same on both ends. Therefore, if it is not known by prior definition, the length of the data block should be passed in the request parameters, retrieved by the user program in #2, and inserted in its Receive Data call.

RECEIVE OR TRANSMIT DATA ONLY

This call, item 2 of Tables A-1 and A-2, will be used to handle the data transfer requested by the Transmit Request and Receive or Transmit Data call.

NOTE

The data buffer length must be exactly the same as the data buffer length in the Transmit Request and Receive or Transmit Data call.

RECEIVE OR TRANSMIT REQUEST ONLY

This call, item 3 of Tables A-1 and A-2 can be used in various ways.

a. A Receive Request call is used to handle the request portion of a Transmit Request and Receive or Transmit Data call generated in a remote computer.

b. A Transmit Request call can be used to pass information contained in the request parameter buffer to a remote computer that would accept the information with a Receive Request call.

c. The Receive Request call is used in the BCS polling mode operation.

ENABLE LISTEN MODE

This call, item 4 of Table A-1 and A-2, enables the software listen mode. The software listen mode or the software polling mode should be established early in the user's program to allow the driver to handle requests from the other computer.

Listen mode allows a user-written BCS subroutine to be executed upon reception of an interrupt from a remote computer. The subroutine to be executed must have an entry point called INT65. Subroutine can be written in Assembler language or FORTRAN. Subroutine can have no arguments or parameters, and it must not call .IOC., any library subroutines that are used anywhere else in the system, or any other non-reentrant subroutine.

Assembler language constraint is that the entry point of INT65 must be followed with a BSS of 2. Another is that before leaving INT65, the return point must be incremented by one, thus returning to P+2.

Subroutine INT65 communicates with D.65 through DIR65 and DRL65. If INT65 is written in Assembler, DIR65 must be included when user programs are loaded. If INT65 is written in FORTRAN, DIR65 and DRL65 must be included at BCS system generation. In Assembler, driver calls are made to DIR65. In FORTRAN, driver calls are made to DRL65.

If more than one computer serial interface card requires a listen mode environment, additional requirements must be met. The primary requirement is that D.65, DIR65 and perhaps DRL65 must be repeated for each computer serial interface card, requiring listen mode. BCS generation does not allow duplicate drivers; therefore, the additional drivers must be entered with new designators. This is done by changing the name of D.65 to D.XX, where XX is 4Ø to 77 and not used by any other driver in the system. The names of DIR65 and DRL65 must also be changed to XX. All entry and exit points must be changed to reflect the new number. Subroutine INT65 must be written as INTXX and its references to DIRXX and DRLXX must use the same number.

The user-written subroutine INT65 will normally read the request, set things in order, and read the transmitted data or transmit data. Since calls to IOC (or any of the standard features of BCS) may not be made (during the running of program INT65), driver D.65 provides call sequences which are direct calls to the driver. These calls are like the Assembler calls to D.65 in Table A-1 but are called by a JSB to DIR65. A FORTRAN example is provided in the description of the sample program at the end of this section. The format is as follows:

```
        NAM     INT65
        NOP
        NOP
        NOP
        JSB     DIR65
        OCT     XX1YY
        JMP     Reject Address
        DEF     BUFF
        DEC     3
        •
        •
BUFF    OCT     IMODE
        DEF     IDBUF
        DEC     IDBFL
        •
        •
        ISZ     INT65
        JMP     INT65,I
```

ENABLE POLLING MODE

The polling mode (item 5 of Tables A-1 and A-2) is available to BCS and D.65 only. This mode allows D.65 to service as many computer serial interface cards as are installed in the computer card cage. In the polling mode, the user's program checks each communication card with a Read Request call. If no request has been received, the driver returns with status bit 6 set denoting "No Request Received". If a request has been received, the driver will handle the request as a normal request transfer.

Figure A-1 provides an abbreviated example of a program which may be used to poll three computer serial interface cards. The example includes only the complete polling of the card in I/O slot 15 and omits program segments where actions taken depend on user/programmer requirements and decisions. A useful addition to the program shown would be a counter which allows the clearing of the device and the re-enabling of the polling on an error to occur a limited number of times before remedial action is indicated.

TRANSMIT STOP REPLY

This call (item 6 of Tables A-1 and A-2) is returned when a request is read and the driver cannot handle it. Possible reasons for transmitting a STOP are:

a. A request parameter is in error.

b. A transmit request and transmit data call is received. For some reason, the driver cannot handle the call. A transmit STOP reply is returned to halt the data transfer at this time.

Table A-1. D.65 Assembler Calling Sequences

1. Transmit Request and Receive or Transmit Data

```
        JSB     .IOC.
        OCT     XX1YY
        JMP     Reject Address
        DEF     BUFF
        DEC     5
          •
          •
BUFF    OCT     IMODE
        DEF     IDBUF
        DEC     IDBFL
        DEF     IRBUF
        DEC     IRBFL
```

2. Receive or Transmit Data Only

```
        JSB     .IOC.
        OCT     XX1YY
        JMP     Reject Address
        DEF     BUFF
        DEC     3
          •
          •
BUFF    OCT     IMODE
        DEF     IDBUF
        DEC     IDBFL
```

3. Receive or Transmit Request Only

```
        JSB     .IOC.
        OCT     XX1YY
        JMP     Reject Address
        DEF     BUFF
        DEC     3
          •
          •
BUFF    OCT     IMODE
        DEF     IRBUF
        DEC     IRBFL
```

Where:

XX = 10 for READ (receive)

XX = 20 for WRITE (transmit): See note.

YY = Device reference number

IDBUF = Buffer for data storage

IDBFL = Data buffer length; a positive number of words

IRBUF = Buffer for request storage

IRBFL = Request buffer length; a positive number of words

IMODE = Two octal digits, AB, where "A" is the data transmission mode and "B" is the driver submode.

A = 0 for DMA open loop (READ and WRITE)

A = 1 for DMA special open loop (WRITE only)

A = 2 for dedicated open loop (READ and WRITE)

A = 3 for interrupt open loop (WRITE only)

A = 4 for interrupt special open loop (READ only)

A = 5 for interrupt closed loop (READ and WRITE)

B = 0 for call sequence "1" above

B = 1 for call sequence "2" above

B = 2 for call sequence "3" above

4. Enable Listen Mode

```
        JSB     .IOC.
        OCT     302YY
        JMP     Reject Address
```

5. Enable Polling Mode

```
        JSB     .IOC.
        OCT     303YY
        JMP     Reject Address
```

6. Transmit STOP reply

```
        JSB     .IOC.
        OCT     301YY
        JMP     Reject Address
```

7. Clear Request

```
        JSB     .IOC.
        OCT     000YY
```

8. Status Request

```
        JSB     .IOC.
        OCT     400YY
```

Where:

YY = Device reference number.

NOTE

The XX of the OCT pseudo instruction is 21 for buffered IOC transmissions and 31 for items 4, 5, and 6. For example, 211YY for item 1 instead of 201YY and 312YY for item 4 instead of 302YY.

Table A-2. D.65 Call Sequences (FORTRAN)

Call	Calling Sequence
1. Transmit Request and Receive or Transmit Data	CALL L65 (IRW, IDVN, IMODE, IDBUF, IDBFL, IRBUF, IRBFL)
2. Receive or Transmit Data Only	CALL L65 (IRW, IDVN, IMODE, IDBUF, IDBFL)
3. Receive or Transmit Request Only	CALL L65 (IRW, IDVN, IMODE, IRBUF, IRBFL)

Where:

IRW = Request code

IRW = 1 for READ (receive)

IRW = 2 for WRITE (Transmit)

IDVN = Device reference number. See note 1.

IDBUF = Buffer for data storage

IDBFL = Data Buffer length; a positive number of words

IRBUF = Buffer for request storage;

IRBFL = Request buffer length; a positive number of words

IMODE = Two octal digits, AB, where "A" is the data transmission mode and "B" is the driver submode.

A = 0 for DMA open loop (READ and WRITE)

A = 1 for DMA special open loop (WRITE only)

A = 2 for dedicated open loop (READ and WRITE)

A = 3 for interrupt open loop (WRITE only)

A = 4 for interrupt special open loop (READ only)

A = 5 for interrupt closed loop (READ and WRITE)

B = 0 for call sequence "1" above

B = 1 for call sequence "2" above

B = 2 for call sequence "3" above

Call	Calling Sequence
4. Enable Listen Mode	CALL L65 (IRW, IDVN)
5. Enable Polling Mode	CALL L65 (IRW, IDVN)
6. Transmit STOP Reply	CALL L65 (IRW, IDVN)
7. Clear Request	CALL L65 (IRW, IDVN)
8. Status Request	CALL L65 (IRW, IDVN, ISTAT, ITRLG)

Where:

IRW = Request code

IRW = 0 for clear request

IRW = 3 for status request

IRW = 4 for enable polling mode

IRW = 5 for enable listen mode

IRW = 6 for transmit STOP reply

IDVN = Device reference number. See note 2.

ISTAT = EQT entry word 2 returned for status call

ITRLG = EQT entry word 3 returned for status call

NOTES:
1. If buffered IOC is used, the priority bit 9 of the IDVN word must be set for WRITE requests.
2. If buffered IOC is used, the priority bit 9 of the IDVN word must be set for items 4, 5 and 8 (IRW = 5, 4, or 3).

```
FTN,A,B,L
C  COMPUTER SERIAL INTERFACE CARDS ARE IN
C  I/O SLOTS 15, 16, 17.  THIS IS A POLLING
C  MODE EXAMPLE.  CLEAR DRIVER AND INITIALIZE
C  CARDS WITH CLEAR REQUESTS-IRW=Ø
       CALL L65(Ø,15B)
       CALL L65(Ø,16B)
       CALL L65(Ø,17B)
C  ENABLE POLLING MODE-IRW=4
       CALL L65(4,15B)
       CALL L65(4,16B)
       CALL L65(4,17B)
C  POLL SLOT 15 WITH RECEIVE REQUEST ONLY
       CALL L65(1, 15B, 12, IRBUF, 3)
C  CHECK DEVICE STATUS
    1Ø CALL L65(3, 15B, ISTAT, ITRLG)
C  IF DEVICE IS BUSY CHECK STATUS AGAIN
    2Ø IF(ISTAT) 1Ø, 3Ø
C  CHECK FOR SUCCESSFUL COMPLETION
    3Ø IF(IAND (ISTAT, 1)) 1ØØ, 1ØØ, 4Ø
C  A STATEMENT LABELLED 4Ø MUST INITIATE APPROPRIATE
C  ACTION.  FOR EXAMPLE, CHECK ITRLG FOR NUMBER OF
C  REQUEST PARAMETERS PASSED.  INTERPRET REQUEST PARAMETERS
C  AND TAKE ACTION.  IF THE CALL IS A REQUEST AND
C  DATA TRANSFER CALL, THE DATA TRANSFER MUST BE DONE
C  OR REJECTED BEFORE POLLING CAN CONTINUE.
     .
     .
     .
C  IF NOT SUCCESSFUL COMPLETION, CHECK BIT 6 OF ISTAT
C  TO SEE IF REQUEST WAS RECEIVED
   1ØØ IF (IAND (ISTAT, 1ØØ)) 12Ø, 12Ø, 2ØØ
C  IF BIT 6=1, NO REQUEST WAS RECEIVED.  A STATEMENT
C  LABELLED 2ØØ SHOULD CONTINUE USER PROGRAM.  IF BIT
C  6=Ø, CHECK BIT 7 FOR BROKEN LINE AND BIT 5 FOR
C  PARITY ERROR
   12Ø IF (IAND (ISTAT, 24ØB)), 13Ø, 2ØØ, 14Ø
C  FOR EITHER ERROR CLEAR THE DEVICE AND RE-ENABLE
C  POLLING
   13Ø CALL L65 (Ø, 15B)
   14Ø CALL L65 (4, 15B)
       GO TO 2ØØ
   2ØØ CONTINUE
C  POLL SLOT 16, THEN SLOT 17 AS FOR SLOT 15
    CALL L65 (1, 16B, 12, IRBUF, 3)
     .
     .
     .
     .
```

Figure A-1. Typical Polling Mode Program

c. A system is too busy to receive or transmit data at this time.

CLEAR REQUEST

This is a standard BCS clear request to terminate any request being processed and clear the driver. This call must be used carefully. The call clears the driver and initializes the I/O card. Communication with the other computer, if in process, will be disrupted. This call also turns off the listen mode or polling mode. This call should be the first call to the driver after computer power turn on.

STATUS REQUEST

The user-written program should check status after the use of any call which results in the transmission or reception of data or request parameters. The calls are items 1, 2, 3, and 6 of Table A-1 and A-2. The status request returns with EQT entry word 2 and 3. The bits of interest are bit 15 and bits 0 through 7 of EQT entry word 2. See Table A-3.

First check that bit 15 is clear (the computer serial interface card is not busy). Status is not updated by the driver until the request is terminated because of completion or because of an error. Then check that bit 0 is set to verify satisfactory completion of the call. The program may then be continued. If bit 0 is clear denoting an unsatisfactory completion, a jump to a user-written subroutine to check the remaining bits is required.

EQT entry word No. 3 is the transmission log. Bit 15 is 1 at all times for this application and must be masked before operating on the transmission log. Bits 0 through 14 is a log of the number of words transmitted. The value is given as a positive integer. The value is stored only when the request is completed, that is, when all data is transmitted or a transmission error is detected.

Table A-3. BCS Status Bit Descriptions

Bit	Set Bit Denotes	Descriptions
Ø	Request Successfully completed.	Call has been completed; no errors detected.
1	Request Not Accepted.	Request buffer length too long for receiving computer or receiving computer not initialized to accept requests.
2	Transmission Mode Incompatible.	Transmitter and receiver data transmission modes are not compatible. This bit is set in request receiving computer only. Sending computer will get Data Transmission Not Initiated bit set.
3	Data Transmission Not Initiated.	Data transmission mode incompatible (request originating computer only) or data transmission refused by STOP reply.
4	Simultaneous Request.	Response to a request was another request. Both transmitter and receiver will show the same status. Both may try again or one may wait, and one try again as determined by user's program.
5	Parity Error.	Eight consecutive parity errors have been detected. Request is terminated. Line may be open or hardware has failed.
6	No Request Received.	Used in polling mode. A receive request call has been made when there was no request to read.
7	Broken line.	Communication line is open or power removed from remote computer. BCS turns device off.

RTE DRIVER DVR65

All data transmissions are accomplished using DMA transfers. The request receiving driver checks data transmission mode compatibility between transmitter and receiver. The Assembler calling sequences are summarized in Table A-4. To make a FORTRAN call possible, a subroutine named DLK65 is provided to set up proper IOC calling sequences. The FORTRAN calls are summarized in Table A-5.

TRANSMIT REQUEST AND RECEIVE OR TRANSMIT DATA.

This call, item 1 of Tables A-4 and A-5, must be used to pass data. The second computer will read the request with a Receive Request Only call and, after examination of the request parameters, will take appropriate action.

If one computer makes a Transmit Request and Transmit Data call, and the other computer after reading the request does not wish to (or cannot) transfer the data at this time, it must reply with a STOP before other calls to the device will be allowed. If any other action is attempted, bit 1, Request Not Initiated, of the status word will be set. When the first request includes a data transfer, the transfer must be accomplished or rejected before other calls to the driver are allowed.

If when reading the request, the driver in the second computer finds that the Receive Request call specified no or too small of a request buffer, the request will be rejected. Status in both computers would be Request Not Accepted.

Table A-4. DVR65 Call Sequences (Assembler)

1. Transmit Request and Receive or Transmit Data

```
        JSB   EXEC
        DEF   *+6
        DEF   IRW
        DEF   IDVN
        DEF   BUFF
        DEF   BUFFL
        DEF   IMODE
        :
BUFF    DEF   IDBUF
        DEC   IDBFL
        DEF   IRBUF
        DEC   IRBFL
BUFFL   DEC   4
```

2. Receive or Transmit Data Only

```
        JSB   EXEC
        DEF   *+6
        DEF   IRW
        DEF   IDVN
        DEF   IDBUF
        DEF   IDBFL
        DEF   IMODE
        :
```

3. Receive or Transmit Request Only

```
        JSB   EXEC
        DEF   *+6
        DEF   IRW
        DEF   IDVN
        DEF   IRBUF
        DEF   IRBFL
        DEF   IMODE
        :
```

Where:

IRW = Request code.

IRW = 1 for READ (receive)

IRW = 2 for WRITE (transmit)

IDVN = Device reference number.

IDBUF = Buffer for data storage

IDBFL = Data buffer length

IRBUF = Buffer for request storage

IRBFL = Request buffer length

IMODE = Two octal digits, AB, where "A" is the data transmission mode and "B" is the driver submode.

A = 0 for DMA open loop (READ and WRITE)

A = 1 for DMA special open loop (WRITE only)

B = 0 for call sequence "1" above

B = 1 for call sequence "2" above

B = 2 for call sequence "3" above

4. Enable Listen Mode

```
        JSB   EXEC
        DEF   *+3
        DEF   RCODE
        DEF   CONWD
        :
RCODE   OCT   3
CONWD   OCT   1YY
```

5. Transmit STOP Reply

```
        JSB   EXEC
        DEF   *+3
        DEF   RCODE
        DEF   CONWD
        :
RCODE   OCT   3
CONWD   OCT   0YY
```

6. Clear Request

```
        JSB   EXEC
        DEF   *+3
        DEF   RCODE
        DEF   CONWD
        :
RCODE   OCT   3
CONWD   OCT   2YY
```

7. Status Request

```
        JSB   EXEC
        DEF   *+5
        DEF   IRW
        DEF   IDVN
        DEF   ISTA1
        DEF   ISTA2
```

Where:

YY = Device reference number

IRW = Request code.

IRW = 13 for status request

IDVN = Device reference number.

ISTA1 = EQT entry word 5 returned for status call

ISTA2 = EQT entry word 4 returned for status call

Table A-5. DVR65 Call Sequences (FORTRAN)

Call	Calling Sequence
1. Transmit Request and Receive or Transmit Data	CALL DLK65 (IRW, IDVN, IDBUF, IDBFL, IRBUF, IRBFL, IMODE)
2. Receive or Transmit Data Only	CALL EXEC (IRW, IDVN, IDBUF, IDBFL, IMODE)
3. Receive or Transmit Request Only	CALL EXEC (IRW, IDVN, IRBUF, IRBFL, IMODE)

Where:

IRW = Request code

IRW = 1 for READ (receive)

IRW = 2 for WRITE (transmit)

IDVN = Device reference number

IDBUF = Buffer for data storage

IDBFL = Data buffer length; a positive number of words

IRBUF = Buffer for request storage

IRBFL = Request buffer length; a positive number of words

IMODE = Two octal digits, AB, where "A" is the data transmission mode and "B" is the driver submode

A = 0 for DMA open loop (READ and WRITE)

A = 1 for DMA special open loop (WRITE only)

B = 0 for call sequence "1" above

B = 1 for call sequence "2" above

B = 2 for call sequence "3" above

Call	Calling Sequence
4. Enable Listen Mode	CALL EXEC (IRW, ICNWD)
5. Transmit STOP Reply	CALL EXEC (IRW, ICNWD)
6. Clear Request	CALL EXEC (IRW, ICNWD)
7. Status Request	CALL EXEC (IRW, IDVN, ISTA1, ISTA2)

Where:

IWR = Request code

IRW = 3 for clear request

IRW = 3 for enable listen mode

IRW = 3 for transmit STOP reply

IRW = 13 for status request

IDVN = Device reference number

ICNWD = Control word with device reference number YY in octal, e.g., 113B

ICNWD = 0YY for transmit STOP reply

ICNWD = 1YY for enable listen mode

ICNWD = 2YY for clear request

ISTA1 = EQT entry word 5 returned for status call

ISTA2 = EQT entry word 4 returned for status call

After the request is read, the user program must now decide its course of action. The user has the option of declining the data transfer (too large, didn't specify size, program doesn't exist) by sending a STOP reply (Status in #1 would show Data Transmission not initiated) or accepting with a Receive Data call.

If the data transfer request is to be accepted, the user makes a Receive or Transmit Data call. At this time the driver in the request receiving computer checks the data transmission mode information received from the initiating computer against the mode information contained in the Receive or Transmit Data call. If it finds the modes incompatible, it will send a STOP reply to the initiating computer and return to the user's program with status showing Incompatible Mode. The driver in the initiating computer will return with Data Transfer Not Initiated bit set.

NOTE

The data buffer length must be exactly the same on both ends. Therefore, if it is not known by prior definition, the length of the data block should be passed in the request parameters, retrieved by the user program in #2, and inserted in its Receive Data call.

RECEIVE OR TRANSMIT DATA ONLY

This call, item 2 of Tables A-4 and A-5, will be used to handle the data transfer request by the Transmit Request and Receive or Transmit data call.

NOTE

The data buffer length must be exactly the same as the data buffer length in the Transmit Request and Receive or Transmit Data call.

RECEIVE OR TRANSMIT REQUEST ONLY

This call, item 3 of Tables A-4 and A-5 can be used in various ways.

a. A Receive Request call is used to handle the request portion of a Transmit Request and Receive or Transmit Data call generated in a remote computer.

b. A Transmit Request call can be used to pass information contained in the request parameter buffer to a remote computer that would accept the information with a Receive Request call.

ENABLE LISTEN MODE

This call, item 4 of Tables A-4 and A-5, enables the software listen mode. This call enables the driver to accept requests from the other computer on interrupts. If this call is made after the driver has been enabled, it will have no effect.

TRANSMIT STOP REPLY

This call (item 5 of Tables A-4 and A-5) is returned when a request is read and the driver cannot handle it. Possible reasons for transmitting a STOP are:

a. A request parameter is in error.

b. A transmit request and transmit data call is received. For some reason, the driver cannot handle the call. A transmit STOP reply is returned to halt the data transfer at this time.

c. A system is too busy to receive or transmit data at this time.

CLEAR REQUEST

The clear request is used to clear busy flags in the device equipment table. If the interrupt scheduled program was aborted while the driver was handling that device, the clear call needs to be executed on that device to allow normal operation to resume. The clear call is also used if the RTE has downed the device; i.e., open line. A suggested method for implementation of the clear call is a special program, activated only from the system console, that issues a clear call to the device. To make this program general purpose, i.e., applicable to all computer serial interface cards in the computer, it could request the device reference number to be input from the system console. The clear call can also be used to disable the listen mode and proceed to a polling mode.

POLLING MODE

DVR65 now has a polling mode capability. The Clear Request is used to disable the listen mode and proceed to a polling mode. This mode allows DVR65 to service as many computer serial interface cards as are installed in the computer card cage. In the polling mode, the user's program checks each communication card with a Read Request call. If no request has been received, the driver returns with status bit 2 set denoting "No Request Received". If a request has been received, the driver will handle the request as a normal request transfer.

STATUS CALL

The call to EXEC for status returns EQT word 5 and EQT word 4. It does not return with the transmission log. If more than one program is using the same device, the status returned may apply to an I/O call made by the other program rather than the I/O call that preceded the status call in this program. To by-pass these areas, another means of obtaining status is provided.

When the RTE returns to a program from an I/O call, it returns with EQT word 5 in the A Register and the transmission log in the B Register. If the program is written in Assembler language, all one needs to do is save the A and B Registers in some storage location.

Table A-6. RTE Status Bit Descriptions

Bit	Set Bit Denotes	Status Description
0	Request Completed.	Call has been completed with no errors detected.
1	Request Not Initiated.	Request not initiated because driver is busy or is currently receiving or replying to a request.
2	Request Not Accepted or Received.	Request buffer length too long for receiving computer or receiving computer is not initialized to accept the request. In polling mode Request not received.
3	Transmission Mode Incompatible.	Transmitter and receiver data transmission modes are not compatible. This bit is set in the request receiving computer only. Sending computer will get the Data Transmission Not Initiated bit set.
4	Data Transmission Not Initiated.	Data transmission mode incompatible (request originating computer only) or data transmission rejected by a STOP reply.
5	Simultaneous Requests	Response to a request was another request. Both transmitter and receiver will show the same status. Both may try again or, one may wait and the other may try again as determined by user's program.
6	Parity Error	Eight consecutive parity errors have been detected. Request is terminated. (Line may be open or hardware has failed.)
7	Broken Line	Communication line is open or power is removed from the remote computer. RTE down the device.

If the program is in FORTRAN, the information can be saved as follows:

```
DIMENSION IREG(2)
EQUIVALENCE (REG, IA, IREG(1)),
  (IB, IREG(2))
```

Locations IA and IB will then contain the status word and transmission log, respectively, then driver calls are made as:

```
REG=EXEC (as required for type of call
      desired)
or    REG=DLK65 (IRW, IDVN, IDBUF, IDBFL,
      IRBUF, IRBFL, IMODE)
```

See Table A-6 for RTE status bits.

Check that bit 0 is set to verify satisfactory completion of the call. The program may then be continued. If bit 0 is clear denoting an unsatisfactory completion, a jump to a user-written subroutine to check the remaining bits is recommended.

DATA TRANSMISSION MODE COMPATIBILITY

When the driver transmits request parameters, the desired data transmission mode is included. The receiving driver stores the request parameters and, when called to transmit or receive data, compares the choice of transmission mode with the previously entered transmission mode for compatibility. If the transmission modes are not compatible, the receiving driver rejects the call.

Data transmission mode compatibility is summarized in Table A-7. A data receiver must be in a data transmission mode which allows it to keep pace with the data transmitter.

Table A-7. Data Transmission Mode Compatibility

1.	DMA Open Loop (WRITE) requires a compatible: a. DMA Open Loop (READ) or b. Dedicated Open Loop (READ).
2.	Dedicated Open Loop (WRITE) requires a compatible: a. DMA Open Loop (READ) or b. Dedicated Open Loop (READ).
3.	Interrupt Open Loop (WRITE) requires a compatible: a. DMA Open Loop (READ) or b. Dedicated Open Loop (READ).
4.	DMA Special Open Loop (WRITE) requires a compatible Interrupt Special Open Loop (READ).
5.	Interrupt Closed Loop (WRITE) requires a compatible Interrupt Closed Loop (READ).

APPENDIX B

CORE MAPS

This Appendix contains the following:

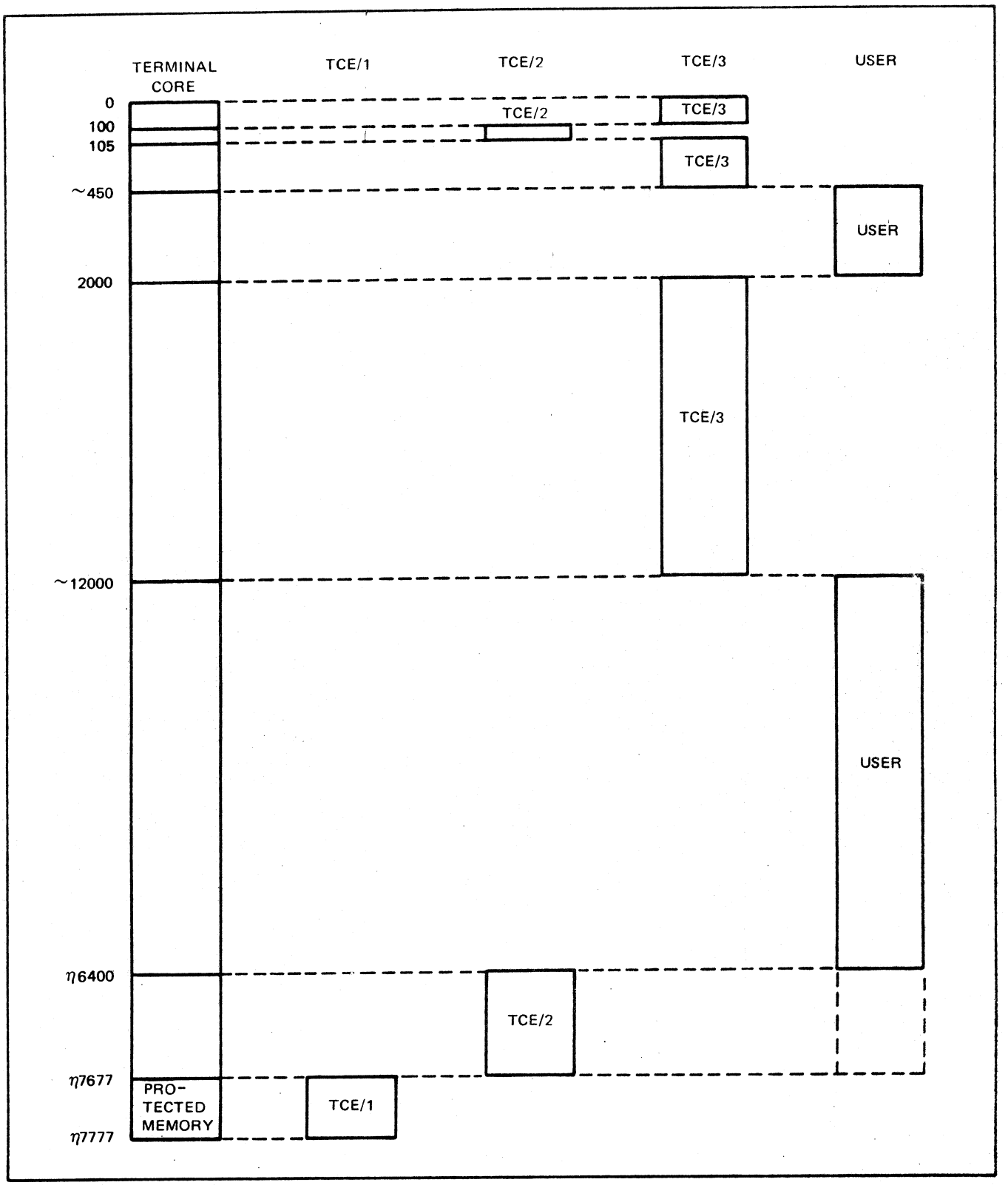


Figure B-1. 91701 Terminal Core Allocation

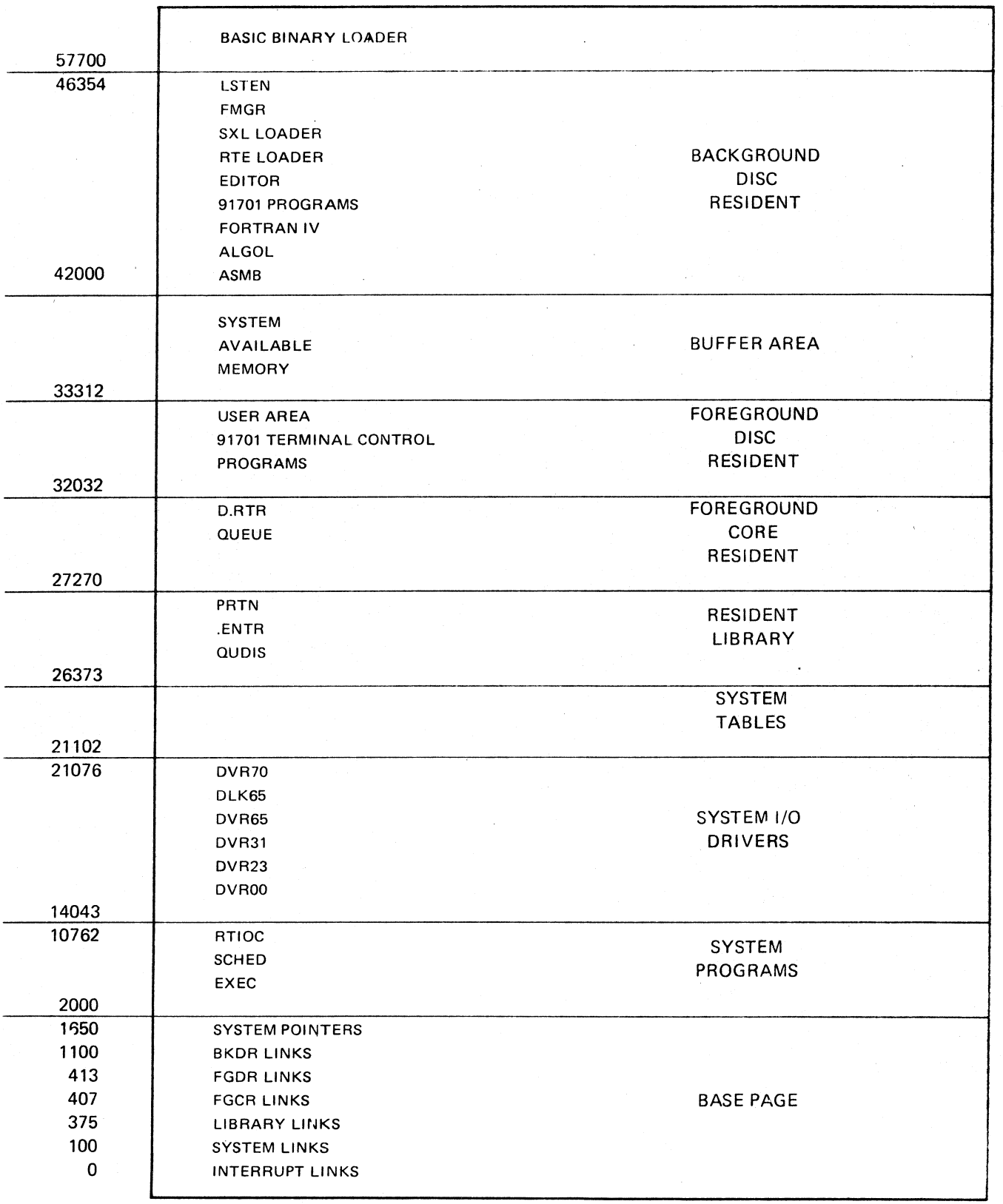


Figure B-2. 24K Central Core Allocation Map

77700	BASIC BINARY LOADER	
77677	LSTEN FMGR SXL LOADER RTE LOADER EDIT FORTRAN IV ALGOL ASMB DLIST	BACKGROUND DISC RESIDENT
47000	PROGL	
	SYSTEM AVAILABLE MEMORY	BUFFER AREA
42235		
	USER AREA 91701 PROGRAMS	FOREGROUND DISC RESIDENT
34204		
	D.RTR ERR TAM DISP QUEUE	FOREGROUND CORE RESIDENT
31067		
		RESIDENT LIBRARY
30172		
		SYSTEM TABLES
21076	DVR70	
17203	DVR65	SYSTEM I/O DRIVERS
16012	DVR31	
15207	DVR23	
14043	DVR00	
10762	RTIOC	
3570	SCHED	SYSTEM PROGRAMS
2000	EXEC	
1650	SYSTEM POINTERS	
1100	BKDR LINKS	
753	FGDR LINKS	
730	FGCR LINKS	BASE PAGE
375	LIBRARY LINKS	
100	SYSTEM LINKS	
0	INTERRUPT LINKS	

Figure B-3. 32K Central Core Allocation Map

APPENDIX C

SYSTEM TABLES

This Appendix contains the following tables:

C-1. RTE Equipment Table (EQT)
C-2. Cartridge Directory
C-3. File Directory

Table C-1. RTE Equipment Table

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

WORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ø
1	DEVICE SUSPEND LIST POINTER															
2	DRIVER "INITIATION" SECTION ADDRESS															
3	DRIVER "COMPLETION" SECTION ADDRESS															
4	D	B	RE-SERVED		S	T	RESERVED			UNIT #			CHANNEL #			
5	AV		EQUIP. TYPE CODE					STATUS								
6	CONWD (CURRENT I/O REQUEST WORD)															
7	REQUEST BUFFER ADDRESS															
8	REQUEST BUFFER LENGTH															
9	TEMPORARY, DISC TRACK #, OR OPTIONAL PARAMETER															
10	TEMPORARY, DISC SECTOR #, OR OPTIONAL PARAMETER															
11	TEMPORARY STORAGE FOR DRIVER															
12	TEMPORARY STORAGE FOR DRIVER															
13	TEMPORARY STORAGE FOR DRIVER															
14	DEVICE TIME-OUT VALUE															
15	DEVICE TIME-OUT CLOCK															

Legend:

D = 1 if DMA is required.

B = 1 if automatic output buffering is used.

S = 1 if a driver is to process time-out on this device.

T = 1 if the device timed out (system sets to zero before each I/O request).

UNIT = fast sub-channel addressed.

CHANNEL = I/O select code for the device (lower number if a multi-board interface).

AV = Available indicator.

Ø = Available for use.

1 = Disabled (down).

2 = Busy (currently in operation).

3 = Waiting for an available DMA channel.

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

ØØ to 07 = Paper tape devices (or system control devices)

00 = Teleprinter (or system keyboard control device).

01 = Photo reader.

02 = Paper tape punch.

1Ø to 17 = Unit record devices.

10 = Plotter.

12 = Line printer.

2Ø to 37 = Magnetic tape/Mass storage devices.

30 = Fixed head disc or driver.

31 = Moving head disc.

4Ø to 77 = Instruments.

Table C-2. Cartridge Directory

The Cartridge Directory is located on the first two sectors of the last track on the system cartridge (LU2). It contains big pointers to all mounted cartridges within FMD.

Sectors	Word	Contents
Ø/1	Ø	LU first cartridge
	1	Last track for FMP
	2	Label word
	3	Lock word
	4	LU second cartridge
		etc.
		⋮
		Ø – end of cartridge spec (up to 31 cartridges)
	124	Ø
	125	Set up (initialized) code word
	126	Set up security code
	127	Spare

Word 125 is the sum of the words in locations 1650 through 1657, and 1742 through 1764.

This order of cartridges can be changed. Refer to the RC Command in the Central Computer operator Requests.

Lock word is either Ø (not locked), or ID segment address of locking program. Only FMGR locks the cartridge in supported software. Locked cartridges are available only to the locker; to others they are considered not mounted.

Table C-3. File Directory

The first entry in Each File Directory is the specification entry for the cartridge itself. Each entry is 16 words long in the format shown below. The directory starts on the last FMP track of each cartridge in sector Ø for all but the system cartridge (LU2), on which it starts in the next logical directory block.

	WORD	
SPECIFICATION ENTRY	0	SIX CHARACTER
	1	INFORMATION
	2	PACK LABEL
	3	LABEL WORD (POSITIVE INTEGER)
	4	FIRST AVAILABLE TRACK
	5	NEXT AVAILABLE SECTOR
	6	NUMBER OF SECTORS PER TRACK
	7	LAST AVAILABLE TRACK FOR FILES
	8	NEGATIVE OR NUMBER OF TRACKS IN DIRECTORY
	9	NEXT AVAILABLE TRACK
	10	FIRST BAD TRACK (OR ZERO)
	11	SECOND BAD TRACK (OR ZERO)
	12	THIRD BAD TRACK (OR ZERO)
	13	FOURTH BAD TRACK (OR ZERO)
	14	FIFTH BAD TRACK (OR ZERO)
	15	SIXTH BAD TRACK (OR ZERO)

Each cartridge has a short label (one word) and a long label. The short label is used for addressing the cartridge; the long label (six ASCII characters), is never used by the FMP and is only used for further ID when listing directories.

The sign bit is set on word 1.

The directory sector address is obtained from the block address by the following formula:

Sector address = (block *14) mode #S/T
where #S/T is the number of sectors per track. Directory blocks are 128 words long.

File entries succeed the specification entry. There is a sixteen word file entry for each file. Words 3 and 7 are different for type Ø files.

Table C-3. File Directory (Continued)

FILE ENTRY

WORD	
0	NAME (CHARACTERS 1 AND 2 OR FILE NAME)
1	NAME (CHARACTERS 3 AND 4 OR FILE NAME)
2	NAME (CHARACTERS 5 AND 6 OR FILE NAME)
3	FILE TYPE
4	TRACK
5	EXTENT/SECTOR
6	NUMBER OF SECTORS IN FILE
7	RECORD LENGTH (TYPE 2 FILES ONLY)
8	SECURITY CODE
9	OPEN FLAG
10	OPEN FLAG
11	OPEN FLAG
12	OPEN FLAG
13	OPEN FLAG
14	OPEN FLAG
15	OPEN FLAG

TYPE Ø FILE ENTRY

WORD	
0	NAME (CHARACTERS 1 AND 2 OF FILE NAME)
1	NAME (CHARACTERS 3 AND 4 OF FILE NAME)
2	NAME (CHARACTERS 5 AND 6 OF FILE NAME)
3	Ø (TYPE)
4	LOGICAL UNIT NUMBER
5	END OF FILE NUMBER
6	SPACING LEGAL CODE
7	READ/WRITE CODE
8	SECURITY CODE
9	OPEN FLAG
≈	
15	OPEN FLAG

Legend:

END OF FILE NUMBER is specified at creation.

Table C-3. File Directory (Continued)

Legend (Continued:

SPACING LEGAL CODE

Bit 15 = 1 Backspace legal

Bit Ø = 1 Forward space legal

READ/WRITE CODE

Bit 15 = 1 input legal

Bit Ø = 1 output legal

If word 1 = Ø the end of the directory reached.

If word 1 = -1, the entry was purged.

The open flags are either zero (not open), or are the ID address of the program the file is opened to.

Each time a file is opened the ID tables for each program that already has the file open are checked to see if the program is dormant. If dormant, i.e., if the point of suspension is zero, or if open to the caller, the open flag is set to zero. This does not close the DCB, nor does it post a possibly unwritten record.

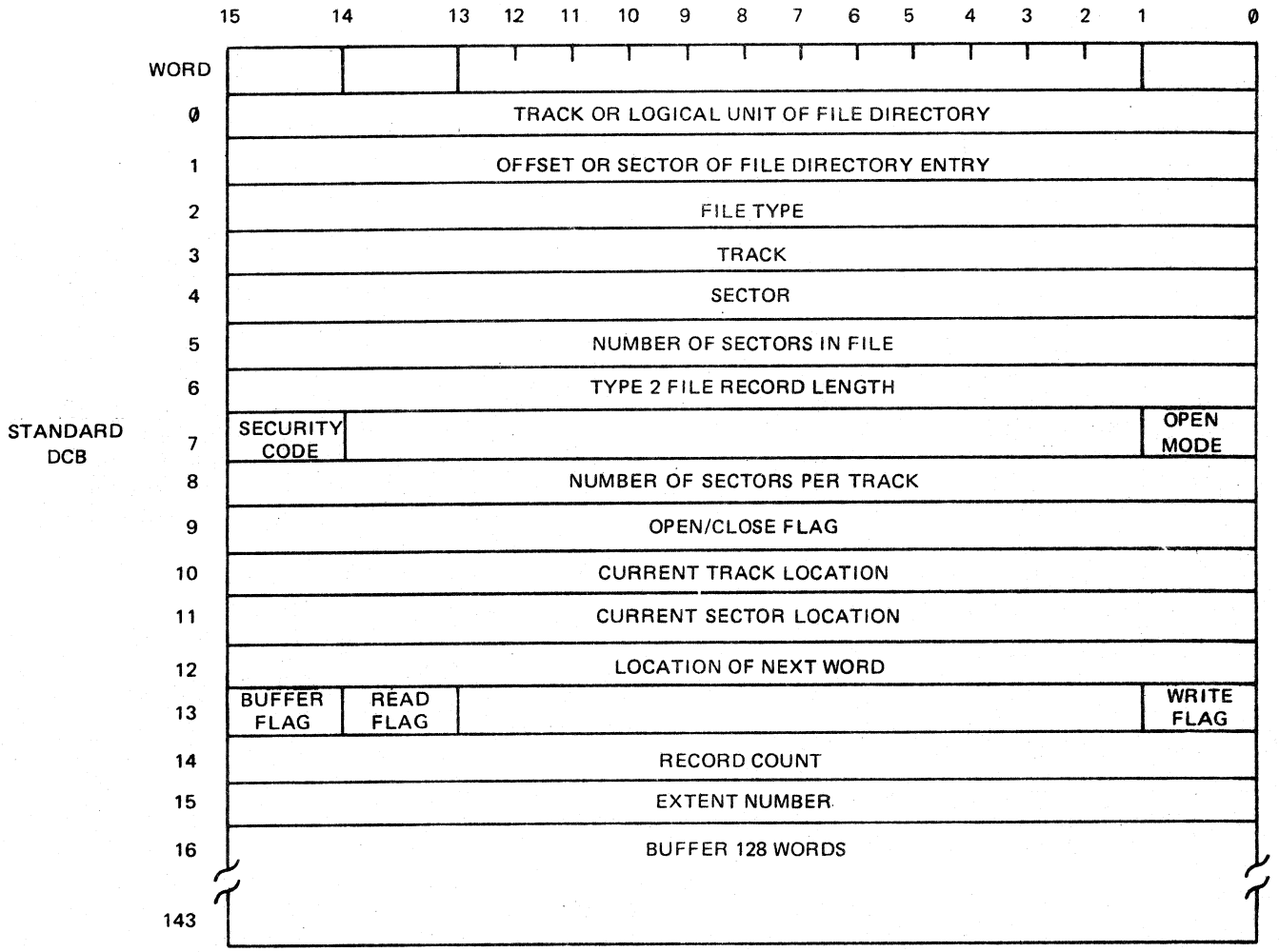
APPENDIX D

RECORD FORMATS AND CONTROL BLOCKS

This Appendix contains the following:

Table D-1. Data Control Block Format

The Data Control Block (DCB) is a 144-word array which the central user must supply and which the 91701 software provides at the terminal. One DCB is required for each open file. The DCB is used to 1) prevent unnecessary directory accesses, 2) keep track of current position within a file, and 3) provide a packing/unpacking buffer. The DCB is free for other use after a PURGE, CLOSE and NAMF call at central. Once a file is open, the DCB is used to reference the file, the name no longer being needed or used.



LEGEND:

SECURITY CODE

1 = Codes Agree
0 = Codes Do Not Agree

OPEN MODE

1 = Update Open
0 = Standard Open

OPEN--CLOSE FLAG

open = ID Segment Address
close ≠ ID Segment Address

BUFFER FLAG

0 = Not In Core
1 = In Core

READ FLAG

1 = If EOF Read
0 = If EOF Not Read

WRITE FLAG

1 = Written On
0 = Not Written On

Table D-1. Data Control Block Format (Continued)

TYPE Ø FILE DCB

WORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ø
Ø	TRACK OR LOGIC UNIT OF FILE DIRECTORY															
1	OFFSET OR SECTOR OF FILE DIRECTORY ENTRY															
2	FILE TYPE															
3	Ø															
4	LOGICAL UNIT NUMBER															
5	END-OF-FILE CODE															
6	LEGAL SPACE CODE															
7	READ/WRITE CODE															
8	NUMBER OF SECTORS PER TRACK															
9	OPEN/CLOSE FLAG															
10	CURRENT TRACK NUMBER															
11	CURRENT SECTOR NUMBER															
12	LOCATION OF NEXT WORD															
13	BUFFER FLAG	READ FLAG														WRITE FLAG
14	RECORD COUNT															
15	EXTENT NUMBER															
16	BUFFER 128 WORDS															
143																

LEGEND:

END-OF-FILE CODE

01xx for Magnetic Tape

10xx for Paper Tape

11xx for Line Printer

NOTE: xx = Logical Unit Number

LEGAL SPACE CODE

1 = Backspace Legal
1 = Forward Legal Space

READ/WRITE CODE

Bit 15 = 1 Input Legal
Bit Ø = 1 Output Legal

There are three basic kinds of disc files:

FIXED RECORD LENGTH FILES

Type 1 and 2 files will be written as presented. They will be packed and may cross sector and track boundaries.

RANDOM LENGTH FILES

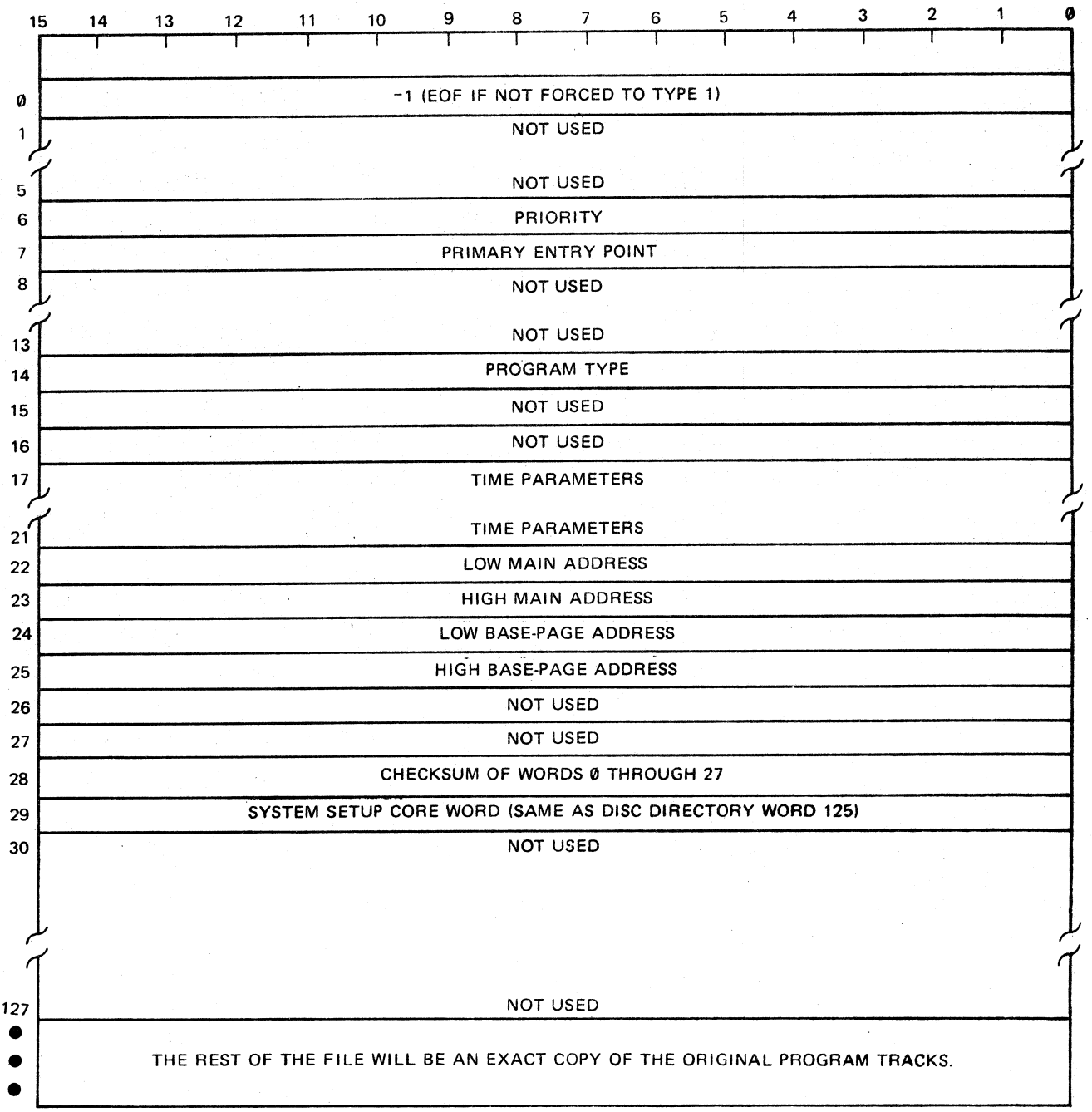
Type 3 and above records will be preceded and followed by a length word which contains the length of the record exclusive of the two length words. A Ø length record will consist of two zero words. An end of file will be indicated by a -1 for the length.

SAVE PROGRAM FILES

Save program files are created by the SP File Manager command as type 6 files; however, they will always be accessed as type 1.

Table D-2. File ID Information Format

The first two sectors of any file are used to store ID information on the program as follows:



APPENDIX E

OCTAL LOADER LISTINGS

This Appendix contains the following:

APPENDIX E

FIXED-HEAD CENTRAL BBDL LISTING

Figure E-1 below is an octal listing of the Basic Binary Disc Loader used for fixed-head RTE configurations. It resides in the protected, highest 64 words of core in the central computer. If the loader is destroyed in core, it can be replaced through the switch register using this listing. The operator simply replaces symbolic items with the value appropriate to the configuration.

A	B 0	1	2	3	4	5	6	7
0*m*7700:	107700	002401	063726	006700	017742	007306	027713	002006
0*m*7710:	027703	102077	027700	077754	017742	017742	074000	077757
0*m*7720:	067757	047755	002040	027740	017742	040001	177757	037757
0*m*7730:	000040	037754	027720	017742	054000	027702	102011	027700
0*m*7740:	102055	027700	000000	006600	1037*cc*	1023*cc*	027745	1074*cc*
0*m*7750:	002041	127742	005767	027744	000000	1*z*0100	0200*nn*	000000
0*m*7760:	107700	063756	102606	002700	1026*qq*	001500	102602	063777
0*m*7770:	102702	102602	103706	1027*nn*	067776	074077	024077	177700

Figure E-1. BBDL Listing

Legend: A + B = Memory Address

- *m* = 1 for 8K, 3 for 16K, 5 for 24K, 7 for 32K memory
- *nn* = first disc channel
- *qq* = second disc channel
- *cc* = photoreader or teleprinter address
- *z* = 6 for 8K, 4 for 16K, 2 for 24K, 0 for 32K memory

MOVING-HEAD CENTRAL BBL LISTING

Figure E-2 below is an octal listing of the Basic Binary Loader used for RTE Moving head disc systems at central, and for non-disc minimally-configured terminal computers. It resides in the protected, highest 64 words of core and is used for paper tape loading.

A	B: 0	1	2	3	4	5	6	7
0*m*7700:	107700	063770	106501	004010	002400	006020	063771	073736
0*m*7710:	006401	067773	006006	027717	107700	102077	027700	017762
0*m*7720:	002003	027712	003104	073774	017762	017753	070001	073775
0*m*7730:	063775	043772	002040	027751	017753	044000	000000	002101
0*m*7740:	102000	037775	037774	027730	017753	054000	027711	102011
0*m*7750:	027700	102055	027700	000000	017762	001727	073776	017762
0*m*7760:	033776	127753	000000	1037*cc*	1023*cc*	027764	1025*cc*	127762
0*m*7770:	173775	153775	1*n*0100	177765	000000	000000	000000	000000

Figure E-2. BBL Listing

Legend: A + B = Memory Address

cc = channel number of Punched Tape Reader

mm = Ø for 4K, 1 for 8K, 3 for 16K, 5 for 24K, 7 for 32K memory

n = 7 for 4K, 6 for 8K, 4 for 16K, 2 for 24K, Ø for 32K memory

TCE/1 LISTING

Figure E-3 below is an octal listing of the TCE/1 used in terminal computers. It normally resides in the protected, highest 64-words of terminal cord replacing BBL. If TCE/1 is distroyed in core, it can be replaced through the switch register using this listing. The operator simply replaces symbolic items with values appropriate to the configuration.

A	B: 0	1	2	3	4	5	6	7
0*m*7700:	1077*cc*	1035*cc*	1025*cc*	067774	017741	067732	017741	007104
0*m*7710:	017741	005323	106401	017741	017753	017753	017753	006021
0*m*7720:	027765	017753	005727	007004	077775	017753	077776	047777
0*m*7730:	017753	002041	177776	037776	067776	037775	027727	017753
0*m*7740:	027714	000000	1066*cc*	1023*cc*	027743	1025*cc*	053772	127741
0*m*7750:	053773	027742	027770	000000	1023*cc*	027754	1065*cc*	1035*cc*
0*m*7760:	002020	002300	063772	1026*cc*	127753	017753	006021	002040
0*m*7770:	102011	102077	170360	007417	170017	000000	000000	1z0100

Figure E-3. TCE/1 Listing

Legend: A + B = Memory Address

m = 1 for 8K, 3 for 16K, 5 for 24K, 7 for 32K memory

cc = remote computer channel

z = 6 for 8K, 4 for 16K, 2 for 24K, ∅ for 32K memory

APPENDIX F

CONVERSION TABLES

This Appendix contains the following:

F-1. Table of Powers of Two

F-2. Octal-Decimal Conversion Table

F-3. ASCII/OCTAL Table

Table F-1. Table of Powers of Two

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

Table F2. Octal-Decimal Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Table F-3. ASCII/OCTAL Table

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
space	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
–	026400	000055

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
.	027000	000056
/	027400	000057
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
↑	057000	000136
←	057400	000137
ACK	076000	000174
①	076400	000175
ESC	077000	000176
DEL	077400	000177
NULL	000000	000000
SUM	000400	000001
EOA	001000	000002
EOM	001400	000003
EOT	002000	000004
WRU	002400	000005
RU	003000	000006
BELL	003400	000007
FE_0	004000	000010
HT/SK	004400	000011
LF	005000	000012
V_{TAB}	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DC_0	010000	000020
DC_1	010400	000021
DC_2	011000	000022
DC_3	011400	000023
DC_4	012000	000024
ERR	012400	000025
SYNC	013000	000026
LEM	013400	000027
S_0	014000	000030
S_1	014400	000031
S_2	015000	000032
S_3	015400	000033
S_4	016000	000034
S_5	016400	000035
S_6	017000	000036
S_7	017400	000037

APPENDIX G

TTY EDIT FEATURES

This Appendix contains the following:

91701 TCE/3 D.ØØD TTY Driver Features

A modified version of the standard BCS D.ØØ TTY driver is used to perform special functions for TCE/3. The calling sequences to .IOC. are the same as for the standard D.ØØ driver. The added features are:

a. Keyboard input can be terminated by pressing carriage return. The driver outputs the line feed. Input of a line feed is ignored.

b. Deletion of an entire input line (buffer load) is accomplished by pressing RUBOUT. The driver outputs a backslash, carriage return and line feed.

c. Deletion of a character is accomplished by pressing BACKSPACE, or by holding the CTRL key down while pressing H. For the latter, the driver outputs a back-arrow.

d. If the last character of an output buffer is a back-arrow, the carrriage return-line feed is suppressed.

e. Operator attention is accomplished by pressing the ESC (Escape) key. The driver may either be doing output, or it is enabled for input; if the latter, there may or may not be a request for input currently pending. If the driver doing output, it is not possible using the standard buffered interface hardware, to identify which character was struck on the keyboard, although the fact that some character was struck can be determined. Therefore, when some key is struck during output, the output is suspended, and the teleprinter thrown into non-echo input. If the next key struck is ESC (Escape), control is passed to the TCE/3 operator prompt seciton; if the next key struck is not ESC, the out= put is resumed (useful to stop the CRT so it can be read). If the driver is not doing output, it is enabled for input. If input has not been requested, characters input other than ESC are echoed but otherwise ignored; the ESC key causes transfer to the TCE/3 operator prompt. If input is requested, characters other than ESC are stored in the requestor's buffer.

f. The printing of an output line may be terminated prematurely by the operator with no effect on subsequent operation. Pressing any key causes the output to stop, as explained above. If the RUBOUT key is then pressed, it will cause the driver to output a backslash, carriage return and line feed. The only effect the caller will see is a reduced character count in the EQT status word.

91701 TCE/2 TTY Driver Features

An abbreviated version of the standard SIO non-interrupt TTY driver is coded in-line with TCE/2. It looks the same to the operator as the TCE/3 BCS D.ØØD driver except for the operator attention (Escape) feature and operator cancellation of output. If the user wishes to use this driver, the calling sequences are as follows:

Input from keyboard:

LDA record size in words (positive)
LDB buffer address
JSB 1Ø4B,I

On return, (B) = # characters input

Output to teleprinter:

LDA record size in words (positive)
LDB buffer address
JSB 1Ø2B,I

Note that although locations 102B and 104B are assembled with driver linkages, they are not used by TCE/2 and are available to user programs for other purposes.

a. Keyboard input is terminated by pressing carriage return. The driver outputs the line feed. Input of a line feed is ignored.

b. Deletion of an entire line (buffer load) is accomplished by pressing RUBOUT. The driver outputs a back-slash, carriage return and line feed.

c. Deletion of a character is accomplished by pressing BACKSPACE or by holding the CTRL key down while pressing H. For the latter, the driver will output a back-arrow.

d. If the last character of an output buffer is a back-arrow, the carriage return-line feed is suppressed.

91701 Operator Commands Syntax for TCE/2 and TCE/3

The keyboard syntax for TCE/2 and TCE/3 is described below.

a. Operator commands may be input when the prompt character (colon (:)) is displayed. This is accomplished by pressing the ESC (Escape) key in TCE/3; in TCE/2, by manually restarting at the TCE/2 start address.

b. Null input and illegal commands cause "SYNTAX ERROR" to be displayed, and another prompt character output.

c. A blank or a comma delimits a command mnemonic (LOAD, RUN, TELLOP, etc.). Imbedded blanks within the mnemonic are not allowed.

d. A comma delimits parameters. Imbedded blanks after the command mnemonic delimiter are ignored unless the parameter is a character string, as in the TELLOP request. Octal parameters must be preceded with an "@" sign. Except for TELLOP, parameters must be positive numeric integers less than or equal to 32767 decimal or 77777 octal.

e. If an error is made in entering the parameters, the rubout and backspace features of the TTY driver may be used.

f. Each request must be terminated by entering a carriage return.

APPENDIX H

HP CHARACTER SET

APPENDIX H

HP CHARACTER SET

RTE SPECIAL CHARACTERS

Mnemonic	Value	Use
SOM (Control A)	1	Backspace
S1 (2600 Backspace) (Control Y)	31	Backspace
EOT (Control D)	4	Simulated End Tape

b_7				0	0	0	0	1	1	1	1
b_6				0	0	1	1	0	0	1	1
b_5				0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1								
0	0	0	0	NULL	DC_0	b	0	@	P		
0	0	0	1	SOM	DC_1	!	1	A	Q		
0	0	1	0	EOA	DC_2	"	2	B	R		
0	0	1	1	EOM	DC_3	#	3	C	S		
0	1	0	0	EOT	DC_4 (STOP)	$	4	D	T		
0	1	0	1	WRU	ERR	%	5	E	U		
0	1	1	0	RU	SYNC	&	6	F	V		
0	1	1	1	BELL	LEM	(APOS)	7	G	W		
1	0	0	0	FE_0	S_0	(	8	H	X		
1	0	0	1	HT / SK	S_1	)	9	I	Y		
1	0	1	0	LF	S_2	*	:	J	Z		
1	0	1	1	V_{TAB}	S_3	+	;	K	[		
1	1	0	0	FF	S_4	(COMMA)	<	L	\		ACK
1	1	0	1	CR	S_5	-	=	M	]		①
1	1	1	0	SO	S_6	.	>	N	↑		ESC
1	1	1	1	SI	S_7	/	?	O	←		DEL

UNASSIGNED

Standard 7-bit set code positional order and notation are shown below with b_7 the high-order and b_1 the low-order, bit position.

	b_7	b_6	b_5	b_4	b_3	b_2	b_1
EXAMPLE: The code for "R" is:	1	0	1	0	0	1	0

LEGEND

NULL	Null/Idle
SOM	Start of message
EOA	End of address
EOM	End of message
EOT	End of transmission
WRU	"Who are you?"
RU	"Are you...?"
BELL	Audible signal
FE_0	Format effector
HT	Horizontal tabulation
SK	Skip (punched card)
LF	Line feed
V_{TAB}	Vertical tabulation
FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
DC_0	Device control reserved for data link escape
DC_1-DC_3	Device Control
DC_4(Stop)	Device control (stop)
ERR	Error
SYNC	Synchronous idle
LEM	Logical end of media
S_0-S_7	Separator (information)
b	Word separator (space, normally non-printing)
<	Less than
>	Greater than
↑	Up arrow (Exponentiation)
←	Left arrow (Implies/Replaced by)
\	Reverse slant
ACK	Acknowledge
ⓘ	Unassigned control
ESC	Escape
DEL	Delete/Idle