Honeywell

SERIES 600

SOFTWARE

GECOS III

TIME-SHARING SYSTEM EXECUTIVE

Honeywell

SERIES 600

GECOS III

TIME-SHARING SYSTEM EXECUTIVE

SUBJECT:

Implementation of the Time-Sharing System Executive for the Comprehensive Operating Supervisor (GECOS III).

SPECIAL INSTRUCTIONS:

This manual completely supersedes the previous edition (CPB-1501A) and includes information published in *Systems Development Letter 3.3.* Technical changes from the previous edition are indicated by change bars.

DATE:

February 22, 1971

DOCUMENT NUMBER:

CPB-1501B

PREFACE

Section I of this manual presents an introduction to the Time-Sharing System. Section II describes the subprograms of the Time-Sharing Executive module. Additional information concerning system macros, the Time-Sharing System deck setup, error messages, and user accounting is given in Appendices A, B, C, and D, respectively.

Additional software maintenance documents for GECOS[1] III are as follows:

GE-600 Line GECOS III Introduction and System Tables, CPB-1488

GE-600 Line GECOS III Startup, CPB-1489

GE-600 Line GECOS III System Input, CPB-1490

GE-600 Line GECOS III Dispatcher and Peripheral Allocation, CPB-1491

GE-600 Line GECOS III Rollcall, Core Allocation, Operator Interface, CPB-1492

GE-600 Line GECOS III Fault Processing and Service MME's, CPB-1493

GE-600 Line GECOS III I/O Supervision, CPB-1494

GE-600 Line GECOS III Error Processing, CPB-1495

GE-600 Line GECOS III Termination and System Output, CPB-1496

GE-600 Line GECOS III File System Maintenance, CPB-1497

GE-600 Line GECOS III Utility Routines, CPB-1498

GE-600 Line GECOS III Comprehensive Index, CPB-1499

GE-600 Line GECOS III Time-Sharing Programmer's Reference, CPB-1514

GE-600 Line GECOS III Basic Language, CPB-1521

[1] Trademark

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

1. INTRODUCTION

The GE-600 Time-Sharing System is one part of a two-part, integrated time-sharing--batch operating system that enables the user to do time-sharing and batch work simultaneously. The structure of the overall system allows variable allocation of resources to each mode of operation so that an installation with an established batch load can add on time-sharing capabilities without major disruption to the batch work. As the time-sharing system load grows, more of the overall system resources can be shifted from batch to the time-sharing work.

This integration of time-sharing and batch systems provides significant capabilities for the user at the remote terminal since the design of the system permits the user to have access to both parts of the system. Using the conversational capabilities of the time-sharing system, a user can prepare input files to be submitted to existing batch programs. The output can be selectively scanned from the remote terminal. When the user is satisfied with the results, a complete printout can be directed to a remote, high-speed printer terminal if desired.

The structure of the time-sharing system (TSS) is such that users are encouraged to add their own processing portion to the system. The general concept is similar to generating a slave program in the batch system. The services provided are directed toward time-sharing and remote terminal requirements. However, the interface is kept general and is designed to permit users or installations to readily develop extensions for their specific needs.

The time-sharing system itself may logically be divided into two categories: the Executive and the subsystem programs. The time-sharing system Executive is a submonitor within the GE-600 Comprehensive Operating Supervisor (GECOS III). The GECOS system performs functions for subsystem programs such as allocation, dispatching, control of I/O, accounting for resources used, etc. The subsystem programs provide functions for the user similar to normal slave programs within the standard operating system.

The executive portion of the system is core-resident (while time-sharing is active), and is made up of a number of subprograms (one of which is named Executive). The various subsystems are brought into the TSS allotted core area as needed to satisfy the user's requests. (The core size is set by the GE-600 central site console operator.) A separate copy of each subsystem is associated with each user, as required (i.e., subsystems are not reentrant). Several subsystems perform executive functions related to user log on and log off.

The overall time-sharing system flow of control is shown in Figure 1. The time-sharing system is a privileged slave program within the GECOS system.

A MME GELBAR is used to dispatch to the subsystems. The return (fault, timer runout) is to a point in the TSS Executive where the cause is determined and appropriate action is taken. In all cases, the IC and I and registers are saved for the subsystem. A derail fault originating from a subsystem is interpreted as a request for service by the TSS Executive. For other fault returns, it is determined if the subsystem will handle faults. If such is the case, return is made to the subsystem (MME GELBAR); otherwise, the fault is interpreted as an error and a message sent to the terminal. However, certain type errors (e.g., parity, lockup, or op-not-complete) are considered as system errors. If a subsystem has exhausted its quantum of time, the Executive is entered to examine the status of remote I/O and up-date status of all users based on this input or output. It then goes through a selection and swapping procedure, subsequently dispatching the processor to a subsystem in core.

A fault occurring within the time-sharing system Executive itself is a program error. In this case, recovery for a limited number of times is attempted before the system aborts. The current state of a subsystem's registers, instruction counters, and indicators is kept within the subsystem's data area. These values are set on initial entry to the Executive and may be used by any of the ensuing processes.

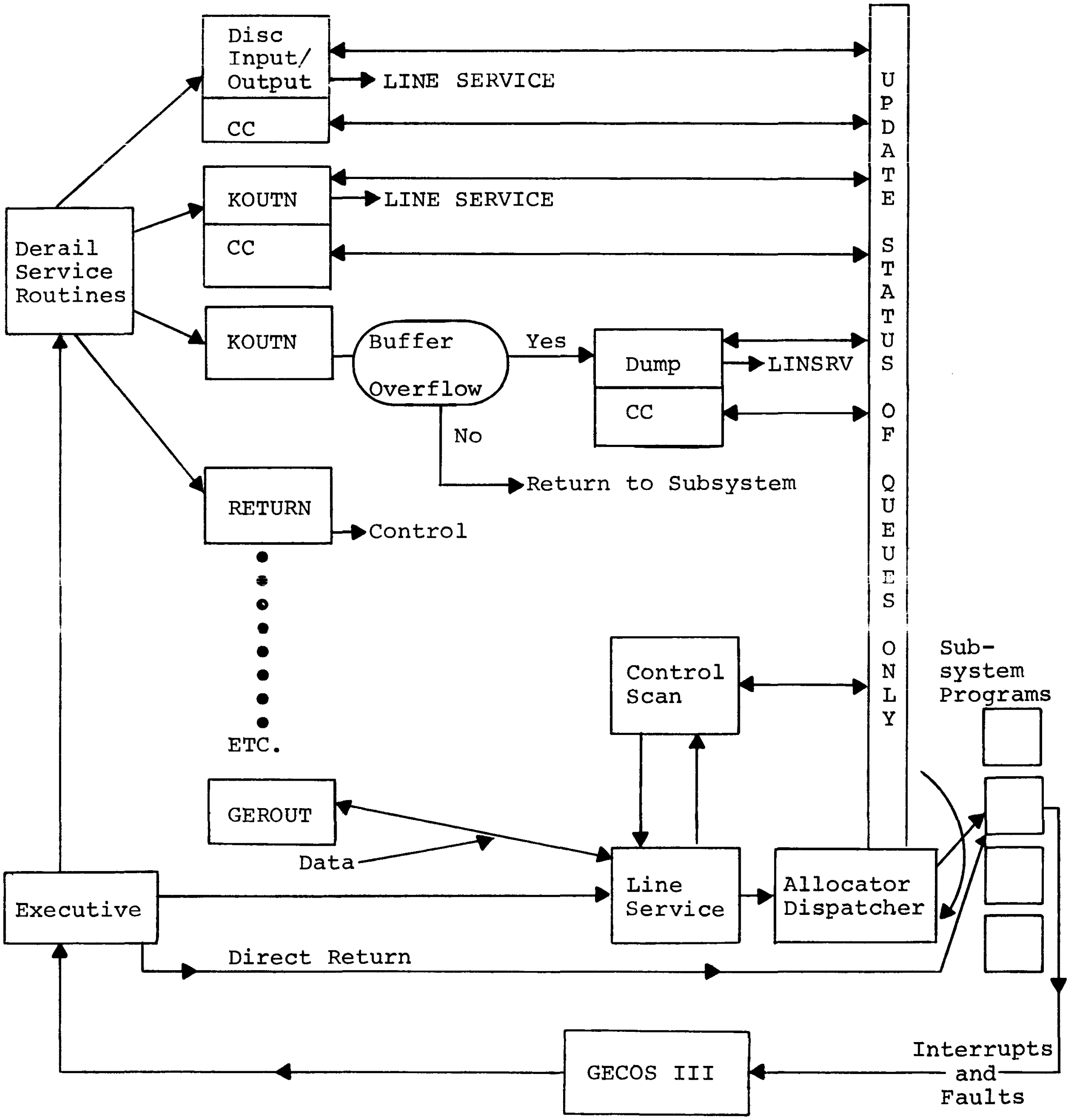


Figure 1. Time-Sharing System Flow of Control

2. SUBPROGRAM DESCRIPTIONS (EXECUTIVE)

The Time-Sharing Executive (.MTIMS) module is composed of subprograms as follows:

●	TSSA	Communication Region
●	TSSB	Trace of Events
●	TSSC	Swap Space Allocator
●	TSSD	PAT Manager
●	TSSE	TSS Exit
●	TSSF	System Error
●	TSSG	Print Error Comments
●	TSSH	Control of User Processes
●	TSSI	TSS Utility Routines
●	TSSJ	Line Service
●	TSSK	Derail Processors
●	TSSL	Allocate Time/Space
●	TSSM	Executive
●	TSSN	UST Generator
●	TSSO	DATANET* 760 and Startup

*Trademark

COMMUNICATION REGION (TSSA)

The Communication Region in the time-sharing system provides a common area in which to store parameters used throughout the system. An effort has been made to collect all storage of parameters (including those within individual routines) into this area. Some of the parameters require initialization at TSS start time; this initialization is provided by TSTART, which loads data into these cells when required at time-sharing startup time. Each program references these parameters from a common communications definition macro which defines the regions with BSS's rather than data definitions. With this procedure, only one deck is required to initialize any of the parameters. If the data layout in the Communication Region is changed, a new Communication Region macro must be defined and each routine reassembled.

UST Assignments

The assignments for the User Status Table (UST) are defined in the Communication Region by EQU pseudo-operations. The location symbols and their contents are shown below. A detailed description of UST appears in the Introduction and System Tables SMD, CPB-1488.

.LID	User ID		
.LSTP	Sum of user processor time		
.LTP	Processor time this interaction		
.LSTIO	No. char/8 to terminal	No. file I/O	
.LTIO	Priority	I/O this interaction	
.LTIN	Time of day this interaction started		
.LSNUB	SNUMB of batch job waiting		
.LBUF	Buffer	B*	T* station ID
.LIOST	I/O status - disc-remote		
.LCC	Courtesy call return		
.LBACK	Backward pointer	priority value	
.LFLAG	Forward pointer	flags	
.LDAT	Current loc of data	bypass count	
.LFILE	Pointer to program stack and AFT		
.LTTYS	Remote status return word		
.LSWTH	Switch word for user		
.LSIZE	Base in TSS	size in core	
.LSWAP	Block No. on drum	actual size	
	Swap courtesy call by user		
.LERRM	Argument to PNTERR	zero	
.LCFIL	Ten-word block for core file		
.LSYBC	*SRC*TAP block count		
.LSTEM	Two-word temp. for Line Service		
.LCALS	CALLSS pushdown list		
.LTALC	Time of day user entered TSS		
.LTPSS	Processor time at subsystem start		

*B = Buffer Tally, 1 bit
*T = Terminal Type, 5 bits

.LKYSS	Key I/O count at subsystem start
.LDCSS	Disc I/O count at subsystem start
.LSPTS	Sum of processor time this subsystem interaction
.LTPS	.LTP at last swap
.LTIOS	.LTIO at last swap
.LTSSV	Saved I/O status
.LFLG2	Second flag word
.LSFTM	4-word LINSRV pointer disc/drum R/W
.LKEYO	Time of Key I/O start
.LSPRT	Sum of processor time in subsystem
.LTRM	Terminate courtesy call
.LINNO	Line number for auto mode
.LINCR	Increment for auto mode
.LTDES	File description of PPT
.LCCTR	DATANET 760 character count
.LBPTR	DATANET 760 buffer pointer
.LSCLP	Line Service previous current line pointer
.LCHG	1-12 character BCI charge number
.LIMIT	Subsystem time limit set by user
.LACPT	Accept indicator for privileged subsystem (Lodx and Cardin)
.LIMTR	Time limit timer

Parameters / Variables

The following are lists of the parameters or variables in the Communication Region with explanations of their functions.

.Lxxxx	Number of items in UST
.LBKMX	Block number threshold for scratch file
.LBUFS	Number of words in line buffer
.LFILS	Max. number of entries in file table
.LPRSS	Number entries in CALLSS pushdown list
.LPRGS	Number entries in pushdown list
.LPRGD	Number entries in program descriptor
.LNTSF	Number of TSS executive files
.LNDRC	Number of entries in DRL count table
.LLNUE	Number of words per UST entry
.LADCW	Max. number swap DCW's
.LNPD	Number of entries in program descriptor list
.LLTOT	Number of new users per cycle
.LLMAX	Max. number of users (180)
.LLWFE	Number of words per file table entry
.LRMAX	Number of system errors allowed before abort
.LSNIO	Number of I/O queues for TSS
.LFPES	TSS user basic PAT entry size
.LFPAT	Number of TSS user PAT's
.LPSZ	Size of subsystem patch area
.LNEVT	Number of entries in trace table
.LBRT	Location of abort, reason code
.LBRK	Break fault

.LSZTM	Size and Time exceed fault
.LBRTM	GELBAR time setting
.LOSTI	Offset for subsystem indicators
.LRAP	Wrapup address
.LFTST	Accumulated fault status
.LOSTR	Offset for subsystem register
.LASTT	Swap I/O status - 2 words
.LADW1	Swap DCW1 and block number cell
.LADW2	Swap DCW2 list
.LASWP	Core map entry, UST pointer for swap in

The list below is ordered according to the location of the cells in the Communication Region block, .TSCOM.

The areas, .TAMAP through .TAREA, are overlayed by MAIN (TSSO) to facilitate reading in large systems such as BASIC and FORTRAN. At the conclusion of startup, these areas are reinitialized.

PARAMETER VARIABLE NAME	SIZE OF CELL	DESCRIPTION
.TCFST	0 (Null)	Origin of common region.
.TCLOC	1	Cell contains location of user's data region and procedure entry at time of actual dispatch to a subsystem.
.TMAT1	1	Temporary storage cell for MAS2 subsystem.
.TESSB	1	Upper portion contains relative base within time-sharing system for current subsystem in operation. The lower portion contains pointer to UST of present user in execution. Set only when passing control to subsystem program.
.TEPTS	1	Contains processor time upon entry to the subsystem program. Used to determine if direct return should be made to subsystem program if Executive regains control before its quanta of time has elapsed. Also used for accounting purposes.
.TELAL	1	When control is given to subsystem program, is set to initial address within time-sharing system of the subsystem. The modifier portion of word contains an index-register-one modification used to reference areas within subsystem program.

.TCLIM	1	Upper portion contains size of subsystem program. Used to check that addresses specified by program are within range of subsystem program. All addresses, calling sequences, etc., are checked before they are used during processing of derails.
.T760	1	Upper portion contains the number of DATANET 760's allowed access to TSS.
.TCPRT	1	Contains processor time upon entry to Executive. Set each time Executive regains control.
.TOSNB	1	Contains original SNUMB for batch jobs. Set at startup time.
.TCDEL	1	Contains incremental limit on time of day for direct return to subsystem.
.TCBSZ	1	Contains in its upper portion size of the line buffers associated with each user.
.TCUST	1	Lower portion contains a pointer to first User Status Table and upper portion contains a pointer to the last User Status Table in use.
.TCPD	1	Upper portion contains a pointer to initial program descriptor. Lower portion contains number of program descriptors present in table.
+1	1	Temporary storage for MAS2 subsystem.
.TLTYN	1	Upper portion contains number of entries that will be processed by Line Service in one cycle. In lower portion, Line Service maintains a pointer for continuation of its scan made during each cycle.
.TLTTM	1	Time of day Line Service last executed remote line terminate scan section of Line Service.
	2	Unused.
.TLOLD	1	Time of day Line Service last executed new user log-on procedure.
.TLNRJ	1	Upper portion contains number of users in reject queue and presently waiting for disconnect.
.TLTLM	1	Time interval used to permit execution of remote line terminate scan in Line Service. (.5 sec.)

.TLNLM	1	Time interval used to permit execution of new user log on procedure in Line Service. (.3 sec.)
.TCNOU	1	Upper portion contains maximum number of users on time-sharing system and lower portion contains the current number of users on system.
.TRCNT	1	Count of the number of serious errors that have occurred in time-sharing system. If count exceeds 5, time-sharing system will be aborted. In general, some recovery is attempted from all system errors but if a series of system errors occurs, whole system is aborted.
.TABFN	1	End-of-allocation queue backwards (0).
.TAFST	1	Upper contains start-of-allocation queue forwards; pointer to first entry in queue.
.TABST	1	Upper contains start-of-allocation queue backwards; pointer to last entry in queue.
.TAFFN	1	End-of-allocation queue forwards (0).
.TAMSZ	1	Contains number of entries on the drum swap map. Justified to the left to be loaded into register 0 as repeat instruction.
.TLFLG	1	Line Service flags. Bit 35, no new jobs can be accepted; bit 34, status request; abort in process.
.TCTM1 . . . .TCTM9	1 each	Series of scratch storage cells used by any subroutine.Not maintained across call from one subroutine to another and must not be used for long term storage data. Not used at courtesy call level or during initial entry to Executive.
.TMAT	1	Temporary storage for MAS2 subsystem.
.TPNTP	1	Temporary storage for Print Error Comments.
.TQSYS	1	Count of times user may have hit break key during log-on.
.TTERM	1	Count of user terminations.
.TQEXC	1	Count of passes through Executive.
.TASIO	1	Count of swaps for keyboard I/O.
.TSSTR	1	TSS startup time.

.TSTRT	1	Count of subsystem starts.
.TKILL	1	Count of subsystem kills.
.TSWAP	1	Count of subsystem swaps.
.TSWPK	1	Sum of swapped core sizes/1024.
.TLUCT	1	Count of users accepted.
+1	1	Count of users rejected.
.TLNAA	1	Count of number of relinquishes because neither the Allocator nor Line Service is busy.
.TLBDC	1	Count of breaks.
+1	1	Count of disconnects.
.TLNCT	1	Count of times alarm clock set when no users.
.TACUT	1	Sum of user log-on time.
.TAURG	1	Number of urgent users.
.TLDID	1	Number of duplicate line ID's.
.TLMON	1	Contains pointer to the master UST in upper and pointer to the UST of the line being monitored in lower.
.TESNB	1	Current SNUMB. Used for batch jobs.
.TCDAT	1	Current date in BCI.
.TDRDR	1	Read command for #S.
.TDRDT	1	Read command for #T.
.TDWDR	1	Write command for #S.
.TDWDT	1	Write command for #T.
.TAUST	4	Used by Dynamic UST module.
.TSDAT	1	Startup date.
.TSTOD	1	Time of day.
.TAAUG	1	Accumulative count of urgent users.
.TLGRF	1	Line Service flag for MME GEROUT.
.TDBLK	1	Block size of #S.
.TDBLT	1	Block size of #T.

.TASSZ	1	Installation-settable subsystem size limitation.
.TASTM	1	Installation-settable subsystem processor time limitation.
.TDSPC	1	Upper portion contains count of MME GEMORE's issued; lower contains count of DRMSPC module calls.
.TDSPL	1	Sum of all #S file links requested via MME GEMORE.
.TDREL	1	Sum of #S file links released.
.TDREC	1	Upper portion contains count of MME GERELS issued; lower contains count of RELSPC calls.
.TDSPD	1	Number of denials.
.TACOR	1	Initial and final address of available TSS core used for swap area.
.TAMPT	1	Forward (upper) and backward (lower) pointers of entries in core map of subsystems loaded.
.TAHOL	1	Upper half contains pointer to first empty slot in core map. If there is request for change in core size, requested size is in lower half.
.TAMAP	2 x maximum number of users	Core map is made up of two-word entries. word 1: 0-17 = pointer to next entry or 0 18-26 = K/2 of core used by SS 27-35 = size of unused hole word 2: 0-8 = initial address of loaded SS 9-17 = index to previous map entry 18-35 = UST pointer When not being used, word 1 contains pointer to next empty slot and word 2 = 0.
.TSMPT	Patch table size + 12	Subsystem patch area constructed by MAST subsystem.
.TSREG	8	Upon entry from GECOS III, Executive stores registers as saved by GECOS III when interrupt occurred. They are then passed on to subsystem in execution or loaded if in executive mode.

.TLNUQ	2 x number users to be processed per cycle	New user queue. New user is held in queue and is either accepted or rejected depending upon status of time-sharing system. Only limited number of users per cycle will be logged onto time-sharing system and each must pass through queue before being accepted or rejected by system.
.TRQUE	2 per entry	Queue contains user error entries that have occurred up to time time-sharing system has been aborted. Upper portion of first word contains UST entry address. Second word contains error code.
.TCCMT	30	Common message buffer.
.TDDRL	maximum number of derails	Series of cells in which count is kept of number of times each derail is called.(.LNDRC table of derail counts.)
.TADMP	2 x maximum number of users plus 1	Map of the drum used to swap user's programs on and off drum. Each map entry contains information in two words. First word contains number of blocks in a region.Lower portion of first word contains the size of residual hole for this block. Upper portion of second word contains initial block number for this region. Lower portion of second word presently not used.
.TADMT	maximum number of users/2	Map for secondary swap file #T.
.TEVTB	event entries-2	Trace table; 2 words per event entry.
.TAREA	1	Pointers for TSSO use, initialization.
.TFMAX	1	Lower portion contains maximum number of users. Set by installation.
.TFPPE	1	Upper portion contains offset address to beginning of PAT pointers in SSA2; lower contains number of PAT entries provided for (initialized by startup).
.TFTEM	1	Device type for temporary user files (bits 0-6).
.TFSY	5	File description of SY**.
.TFSKL	10	Area used by file routines to build PAT's.
.TFBLK	(maximum numbers of users)/6 + 2	SY** assignments.

.TSTRC	1	Entry to the trace event routine.
.TEVNT	2	Tally and refresher tally for the trace table (.LNEVT).
+2	2 x number of event entries-2	Trace table; 2 words per event entry.
.TEVMK	2	Mask of events to be traced (1) or not traced (0).
.TCTPT	20	Table of interaction counts versus processor time intervals.
.TCTSZ	20	Table of interaction counts versus program size.
.TCTDC	20	Table of interaction counts versus number of disc I/O's performed.
.TDSKS	1	Seek command for #S.
.TDSKT	1	Seek command for #T.
.TATSZ	1	Number of entries in #T swap map.
DVTB1	6	Device table 1 is used by TSS startup when space is not available as specified in .TCFIL. Space will be acquired from the first device in this table which has room.
DVTB2	6	Device table 2 is initialized at startup so that any device which does not exist is set zero. File space for SY** and other user's temporary files is acquired on the first device in this table which has room.
.TCFIL	2 x number of TS files	TSS system files. word 1: 0-17 = offset to PAT pointer 24-35 = file ID in BCI word 2: 0-5 = device type 6 = lock bit 7 = 0/1 or temp/perm 9-17 = no. links/file 18-35 = no. words/block
	1	Unused
.TCLST	0	End of communication region.

Program Descriptor Region

The TSSA module also contains a Program Descriptor area in block .TPRGD. This area is physically divided into two sections: the program descriptor and the command language primitive list.

The program descriptor format consists of nine words in the following order:

1. Subsystem name in ASCII
2. Program size, load size
3. Entry point, parameters
4. Seek address (#P), IA load
5. Command language pointer, number of words in CL
6. Program statistics
7. Sum of processor time
8. Sum of disc I/O counts
9. Sum of key I/O counts

The Program Descriptor area is described in detail in the GE-600 Line GECOS III Time-Sharing Programmer reference manual, CPB-1514.

TRACE OF EVENTS (TSSB)

The Trace of Events (TSSB) updates the system trace table and gathers accounting statistics which are not entered in the table.

TSSB contains one entry point:

- EP1 Trace Recent Events (TSEVNT)

TSSB occupies approximately 330 (octal) core storage locations.

TRACE RECENT EVENTS

TSEVNT (EP1 of TSSB) maintains a trace of the most recent events of the time-sharing system and also gathers statistics without making entries in the trace table.

CALLING SEQUENCE

TSEVNT is called from TERM, TRMCU, KEYIO, SYS, STARTP, SCAN1, SCAN3, LINSRV, GOOD10, STG76, EXDRL, SYSRET, ALLOC, ALLOCI and SWAP via:

```
8         16
EVENT     event number
```

which generates the code:

```
XEC       .TSTRC
ZERO      name,0
```

.TSTRC contains:

```
XED       TSEVNT
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to calling program.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The trace table is a 2-word entry, circular table. Two tally words are used to drive it, a current tally word and a refresher tally word. These words and the table are in the Communication Region.

The format of the table entries is:

word 1	Available to user		
word 2	Available to user	User No.	Event No.

However, the user-provided code must do this. A routine in TSEVNT, USER, computes the user number by sequence in UST storage, and moves it to the proper position in the Q. Then the user can provide the event number and whatever else is desired in the A and Q. Another routine, STORE, puts the AQ in the table as a 2-word entry and refreshes the tally if necessary before returning to the time-sharing system.

A 2-word mask (.TEVMK) is maintained against which a test can be made of up to 72 events. A zero in the mask inhibits the trace on that event number. The mask can be modified from the master terminal.

The following table lists the EVENT's and the corresponding trace entries which exist in the current system.

EVENT No.	Source	Word 1		Word 2		
		0 17	18 35	0 17	18 26	27 35
1	TERM	XR1	.LFLAG bits		user	1
2	KEYIO	XR1	I/O Mode		user	2
3	SYS	XR1		Station Code	user	3
4	STARTP	ASCII Name		XR1	user	4
6	LS	TOD		No. of users		6
7	LS (new user)	TOD			user	7
8	SCAN	ASCII Command			user	10
9	DRL	DRL op		DRL Loc	user	11
10	ALLOC (START)				user	12
11	ALLOC (KILL)				user	13
12	ALLOCI (New)	C(.LSIZE)			user	14
13	ALLOCI (Old)	C(.LSIZE)			user	15

The following EVENTs gather statistics on the system but do not make entries in the trace table.

14	KEYIO	Collect interaction counts.
16	SCAN,DRL	Collect subsystem usage.
17	ALLOCI	Accumulate swap count and swap sizes.
18	TERM	Accumulate users' log-on time.

Programming Method

TSEVNT is nonreentrant and relocatable.

Interrupts are inhibited throughout.

Storage

No internal temporary storage is used.

Other Routines Used

Provide Date and Time of Day FGTIM (EP8 of .MFALT)
Obtain Normal Snapshot GESNIP (EP1 of .MSNP1)

SWAP SPACE ALLOCATOR (TSSC)

The Swap Space Allocator (TSSC) manages the swap space on the #S and #T files. It contains three entry points.

- EP1 Request Drum Space (DRMSPC)
- EP2 Return Drum Space (RETSPC)
- EP3 Release #S Links (RELSPC)

The drum map (.TADMP) consists of two-word entries as follows:

Word 1	Number of blocks in use	size of hole
Word 2	Initial block number	zero

TSSC occupies approximately 510 (octal) core storage locations.

REQUEST DRUM SPACE

DRMSPC (EP1 of TSSC) searches the #S map to find a hole to fit the request with the least amount of waste space. Unless it is at the end, entries following the hole space are moved down to create a new entry in the proper location in the map. If space is not available, an attempt to increase the size of #S is made with GEMORE.

If this is unsuccessful, space is found then on #T, the secondary swap file. Normally #S is on the fastest device available and #T is on a less desirable device. Therefore, only as a last resort is #T used.

PRECALLING SEQUENCE

Prior to entering DRMSPC, the registers listed must contain the data indicated.

AU   Size required (in words)

CALLING SEQUENCE

DRMSPC is called from ALLOC, ALLOCI, ALL7AC and DYUST via:

```
8         16
TSX1      DRMSPC
Error return
Normal return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

DRMSPC returns to the calling program. Error return indicates no room available.

POSTCALLING SEQUENCE

AU contains initial block number, if normal return.

SUPPORTING INFORMATION

Programming Method

DRMSPC is nonreentrant and relocatable.

Interrupts are inhibited throughout.

Storage

No internal temporary storage is used.

Other Routines Used

Communication Region (TSSA)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Additional Mass Storage Request MINK (EP2 of .MMORE)

RETURN DRUM SPACE

RETSPC (EP2 of TSSC) searches .TADMP for the matching initial block number. If not found, the request is ignored. When found, the hole of previous entry is adjusted and the entry removed from the map by moving all the following entries up.

PRECALLING SEQUENCE

Prior to entering RETSPC, AU must contain the initial block number returned by DRMSPC or no action will be taken.

CALLING SEQUENCE

RETSPC is called from ALLOC and ALL7AC via:

```
8          16
 TSX1      RETSPC
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

RETSPC returns to the calling program.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RETSPC is nonreentrant.

Interrupts are inhibited until processing is complete.

Storage

Internal temporary storage is used for saving registers and working storage.

Other Routines Used

None.

RELEASE #S LINKS

RELSPC (EP3 of TSSC) examines the size of #S in order to release unneeded links to the system. When hole space exceeds ten percent of the total size of #S, as many links as possible are released from the end of the file via MME GERELS. The minimum retained for #S is the number of links assigned at startup initialization.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

RELSPC is called from LINSRV via:

```
8         16
TSX1      RELSPC
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

RELSPC returns to the calling program.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RELSPC is nonreentrant and relocatable.

Interrupts are inhibited throughout.

Storage

No internal temporary storage is used.

Other Routines Used

MME GERELS Processor RELS (EP1 of .MRELS)

PAT MANAGER (TSSD)

The PAT Manager (TSSD) manages the user's Available File Table (AFT) and Peripheral Assignment Table (PAT) entries in the Slave Service Area (SSA). It provides a common routine that can be used by the various File I/O derails to maintain each user's threaded list of open files. The combined AFT and PAT entries representing an open file in a user's list is called a "file entry."

TSSD contains three entry points.

- EP1 File Entry Space Allocation (PATMAN)
- EP2 File Deallocation (PATDEA)
- EP3 Delinking a File (FREE)

TSSD occupies approximately 510 (octal) core storage locations.

TSSD manages either most or all of the space in SSAs 2-7, depending upon whether any I/O queues (additional to those in SSA1) exist at the beginning of SSA2, i.e., high-order end.

In either case, the area managed by TSSD is essentially divided into two sets of discrete spaces. One set is a single-chained list of available spaces (usually scattered throughout the whole area) called the "available space chain". The high-order word of each variable-size element of the chain describes the number of words available in the element (going towards low-order memory), including itself, in its upper half; and contains a pointer to the next element in the chain in its lower half. The origin of this chain is the lower half of the first word of PATMAN's area, which cell is pointed to by the Communications Region cell .TFPPE, upper. (All pointers in this discussion are negative offsets from the LAL of TSS.)

The second set of spaces is a collection of user's open-file lists, each list consisting of from 1 to (nominally) 20 file entries double-chained together. Each list is open, the backward and forward pointer at either end of the list being null (0), and the "first" entry in each list (backward pointer null) is pointed to in the lower-half of the file-list pointer word in each corresponding UST. (Bits 12-17 of this word contain a count of files currently open for that user, i.e., the current number of entries in the list pointed to.) This word is pointed to in turn by .LFILE lower, also in the UST.

The size of each file entry varies from 8 to 16, according to current practice. The upper limit on the number of entries in a file list, 20 as released, is an assembly parameter that is specified in .LFILS, in TSSA.

TSSD manages the transfer of elements from one set of spaces to the other. It transfers from the available-space chain to a user's file list (PATMAN entry) or vice-versa (FREE), besides releasing or deaccessing files (PATDEA).

FILE ENTRY SPACE ALLOCATION

PATMAN (EP1 of TSSD) determines the size of a new entry and obtains a pointer to the user's file list from the current UST. It is used by several file-derail processors, subsequent to a call to the file system or .MALC9 to access a permanent file or obtain temporary file space. PATMAN attempts to find sufficient space in the second area dedicated to user's PATs, and if successful, links the combined file entry passed by the caller into the user's threaded file list, and returns an offset to the PAT pointer. In some cases PATMAN is called following an extension of a temporary file to relink a previously freed entry (see FREE). It is also called during logging-on to initiate the user's file list with an SY** entry.

PRECALLING SEQUENCE

None

CALLING SEQUENCE

PATMAN is called from SYMAN, DRLFIL, TASK, FILACT and MORLNK via:

```
8       16
TSX1    PATMAN
Zero    L(pate),0
Error return--duplicate name
Normal return
```

(pate--the combined AFT/PAT entry)

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

PATMAN returns to the calling program.

POSTCALLING SEQUENCE

Error return:

X5 = 0, no entry space available
X5 ≠ 0, new file name duplicates the file pointed to

Normal return:

X5 = Offset to entry
Count of open files in UST is updated

SUPPORTING INFORMATION

On allocation, PATMAN determines the size of the new entry, and obtains a pointer to the user's file list from the current UST. It compares the new file name with all files already in the list (if any), rejecting any duplicately-named new entry. Then PATMAN cycles through the available-space chain to find a space as large as, or larger than, the new entry.

If the first such space found is larger than needed, the size field of the available-space chain word is adjusted accordingly and the word is moved to above the new entry space, entailing readjustment of the preceding chain word. If the space just fits, the preceding chain word is modified to point to the succeeding one, and the new entry is copied into the space so obtained.

The "old" end of the user's file list is found (if any), and the new entry is made the "new" end of the list, by manipulation of the respective forward and backward pointers, and the offset to the new entry is stored for return to the caller in X5. The PAT-pointer field, bits 4-17, of the first word of the file entry is set to point to the PAT body (always self-relative + 3), the user's open-file count is incremented, and return is made to the caller.

Programming Method

PATMAN is nonreentrant and relocatable.

Interrupts are inhibited throughout.

Storage

Internal temporary storage is used for saving registers and working storage.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

FILE DEALLOCATION

PATDEA (EP2 of TSSD) removes the file entry and releases the file space (temporary file) or deaccesses the file (permanent file), either for one specified file or for all of the user's open files (excepting SY**). PATDEA is also called by TSSI to deallocate all of the user's files, again excepting SY**.

PRECALLING SEQUENCE

Prior to entering PATDEA, the registers listed must contain the data indicated.

AQ	File name in ASCII or 777 octal in AL if all files in user's AFT are to be deallocated
X2	UST pointer

CALLING SEQUENCE

PATDEA is called from TSFIN, TERM and DRLRFL via:

```
8         16
TSX1      PATDEA
ZERO      Courtesy call, buffer
Error return
Normal return
Courtesy call pending return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The error return is taken if the specified file is not found, or if no files are found.

The normal return is taken when PATDEA has successfully completed deallocation. The user's open-file count in the file-list pointer word is decremented on return.

The Courtesy Call Pending return is taken following the MME GEFSYE issued by PATDEA (for permanent files) and prior to the courtesy call resulting therefrom. The courtesy call address is only pertinent when a permanent file is to be deallocated (because the file system is called to deaccess permanent files).

The buffer must be a BSS of 380 words. It need only be given when a permanent file is to be deaccessed. It is required by the GECOS III file system.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

If a temporary file, the necessary parameters are obtained, .MALC9 is called to deallocate the space, and the file entry is freed. If a permanent file, the offset to its entry is stored in a MME GEFSYE calling sequence. (The entry is not unlinked at this point.) Note that the entry for SY** is bypassed. If no entries exist, the error return is taken immediately. Otherwise, if "all-files" is specified, a loop is set up to cycle through the complete file list.

When the file list is exhausted and permanent files exist, the MME GEFSYE call for deaccessing the file(s) is set up in the beginning of the buffer (buff), the Allocator is notified that the calling subsystem is doing I/O and is not to be swapped, and the MME GEFSYE is executed from buff. The immediate caller (generally DRL RETFIL) is then returned to via the courtesy call pending return, where control can relinquish to some other portion of the Executive, normally Line Service. When the courtesy call is honored, the still-linked file entries are freed and the normal return is taken.

If one file has been specified, rather than all, the procedure outlined above is followed except that the file list is searched through once for the named file. If it is not found, the error return is taken.

Programming Method

PATDEA is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Return LINKS/LLINKS to the Pool ONE (EP1 of .MALC9)

DELINKING A FILE

FREE (EP3 of TSSD) unlinks a file entry and makes that entry space available.

PRECALLING SEQUENCE

Prior to entering FREE, the registers listed must contain the data indicated.

X2 UST address.
X4 Absolute address of file entry.

CALLING SEQUENCE

FREE is called from FILACT, TASK, TERM and DRLRFL via:

```
8       16
XED     FREE,5
Return
```

from PATDEA:

```
XED     FREE,$
Return
```

The call must be made in master mode.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

POSTCALLING SEQUENCE

Open-file count in file list pointer word in UST is decremented. Space occupied by the entry is zeroed during unlinking process.

SUPPORTING INFORMATION

The entry is unlinked from the list by storing its back pointer into its successor's back pointer, and storing its forward pointer into its predecessor.

The unlinked entry is zero-filled, its length being ascertained in the process. Then, if an available-space element exists below the zeroed space, the number of freed words are added to its count. If not, the zeroed space is linked into the available space chain at its starting point, i.e., is pointed to by the origin word.

The user's file count is decremented prior to return to caller.

Programming Method

FREE is nonreentrant and relocatable.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

TSS EXIT (TSSE)

The TSS Exit (TSSE) performs the wrapup procedures necessary when the time-sharing system is aborted.

TSSE contains one entry point:

- EP1 TSS Wrapup (TSFIN)

TSSE occupies approximately 130 (octal) core storage locations.

TIME-SHARING WRAPUP

TSFIN (EP1 of TSSE) deallocates all the TSS associated files in SSA1 and SSA2 when the time-sharing system is aborted.

PRECALLING SEQUENCE

At TSS startup time, octal location 27 contains the address of TSFIN.

CALLING SEQUENCE

TSFIN is called from the .MBRT2 module.

Upon return from the call to LINSRV, entry is via

```
TRA       WRAPUP
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

When processing is complete a MME GEFINI is executed thus terminating TSS execution.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

When the time-sharing system is aborted, GECOS III gives control to the address in location 27 (octal) as part of the abort procedure. TSFIN sets the NONEW and ABORT bits in .TLFLG and then sends disconnects to all active lines. The disconnects are then handled in the normal way by LINSRV, thus clearing up all UST's and deaccessing (perm) or releasing (temp) any activie files. Control is then returned to TSFIN. TSFIN calls .MALC9 to release the collector file, calls GERELS to release the temporary TSS system files, and calls GEFSYE to deaccess the permanent TSS system files (if any exist). TSFIN then exits with a MME GEFINI. This gives control back to GECOS III, where the abort procedure will be completed.

Programming Method

TSFIN is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

MME GERELS Processor RELS (EP1 of .MRELS)
MME GEROUT Processor GROUT (EP1 of .MROUT)
End Courtesy Call ENCC (EP5 of .MDISP)
Deaccess File FSINT1 (EP1 of .MFS04)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Terminate MME GEFINI Entry FINI (EP2 of .MBRT1)
Line Service LINSRV (EP1 of TSSJ)
Return LINKS/LLINKS to the Pool ONE (EP1 of .MALC9)

SYSTEM ERROR (TSSF)

The System Error subprogram (TSSF) processes all system errors.

It contains one entry point:

- EP1 System Error (ERROR)

TSSF occupies approximately 330 (octal) core storage locations.

SYSTEM ERROR

ERROR (EP1 of TSSF), depending on the type of error, may snap the TSS and resume operation when a recoverable error occurs. If more than the allowable number of errors has occurred, it issues a message to the user, disconnects the terminal, and issues a message to the GE-600 console.

The message at the GE-600 is:

```
*TSS AUTOMATIC ERROR RECOVERY  XXXX
```

At the user's terminal, it is:

```
YOU HAVE BEEN TERMINATED

TERMINATE CODE XXXX
```

In both cases, XXXX is an error code from the table which follows. No messages are issued when TSS aborts (that is, no error recovery exists or recovery has been exhausted). If TSSF is entered via courtesy call, the error is tabulated but the user is not terminated and no snaps are taken.

PRECALLING SEQUENCE

Prior to entering ERROR, the registers listed must contain the data indicated.

Q	4-character error code (9-bit ASCII)
X2	UST address

CALLING SEQUENCE

ERROR is called from LINSRV, EXENTR, ALLOC, ALL7AC, PNTERR and SWAP via:

```
8          16
TSX1       ERROR
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

ERROR returns to the caller following a snap unless entered via courtesy call, in which case TSSF executes a MME GEENDC.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Five errors cause abort unless otherwise stated; SNAP occurs unless otherwise stated.

CODE	SOURCE	MEANING	RECOVERY ACTION
OPFT	EXEC	ILLEGAL OP CODE	ABORT ON 2nd OPFT
MEMF	EXEC	Memory fault	ABORT ON 2nd MEMF
FCDF	EXEC	Fault code fault	Normal
DCFT	EXEC	Divide check fault	ABORT on 3rd DCFT
OVFF	EXEC	Overflow fault	ABORT on 3rd OVFF
760X	STARTUP	760 Buffer lost	Normal
DRLF	EXEC	Derail fault	Normal
SYFT	EXEC	Lockup,ONC, Parity fault (user)	ABORT

TSOU	ALLOC	Allocation queue error	Normal
USTF	LINSRV	Error in UST scan	No disconnect, no user message
HANG	ALLOC	Cannot terminate user I/O	ABORT on 2nd HANG
NUQF	LINSRV	No room in new user table	No snap, no disconnect, no user message
GBSY	LINSRV	DATANET 30 BUSY Status	No snap, no user message
ERRP	PNTERR	Error while issuing error message	Normal
KILL	ALLOC	Attempt to return 0 swap space	ABORT
ALLQ	ALLOCI	Duplicate UST pointer in core map	ABORT on 3rd ALLQ
ALL7	ALLOCI	Request to clear swap space location 0	ABORT on 3rd ALL7
ALLP	ALLOCI	Illegal parameter to ALLOC	ABORT
ALLC	ALLOCI	UST last from core map	ABORT on 3rd ALLC
CUST	CUST	Error in UST COLLECTION	ABORT on 2nd fault
(X1) CM	ALLOC	Incorrect core map	ABORT

Programming Method

ERROR is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

MME GEINOS Processor INOS (EP5 of MIOS)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Update Allocator Queue ALLOC (EP1 of TSSL)
Terminate Error Entry FALT (EP3 of .MBRT1)
Obtain Normal Snapshot GESNIP (EP1 of .MSNP1)
Initiate Remote I/O KEYIO (EP2 of TSSI)
MME GEROUT Processor GROUT (EP1 of .MROUT)
End Courtesy Call ENCC (EP5 of .MDISP)
Send "Stop Paper Tape" to DATANET PPTOFF (DRL STOPPT, TSSK)

EXECUTIVE ERROR MESSAGES (TSSG)

The Executive Error Messages routine (TSSG) prints a preset error message onto a remote device.

It contains two entry points:

- EP1 Print Error Message (PNTERR)
- EP2 Special Entry (PNTTSX)

TSSG occupies approximately 1320 (octal) core storage locations.

Message Text

The messages printed by TSSG are as follows:

Comment Number	Text
1	001 - Incorrect Primitive
2	Bad File I/O Command
3	003 - Bad DCW
4	004 - Address Out of Range
5	005 - Bad DRL Code
6	006 - Level of Control Too Deep
7	007 - Bad Program Descriptor
8	008 - Loop In Primitives
9	009 - System Unknown
10	010 - Program Too Large To Swap
11	011 - Incorrect Core File Usage
12	012 - Program Not Allowed Use of This I/O
13	013 - DRL Allowed Only by LOGON
14	DRL RELMEN - Request Release of Too Much Core
15	015 - Cannot Reset ID
16	016 - Overflow Fault
17	017 - Illegal Op Code
18	018 - Memory Fault
19	019 - Fault Tag Fault
20	020 - Divide Check Fault
21	021 - Bad Status Swap Out #S
22	022 - Bad Status Swap In #S
23	023 - Bad Status Load #P

24	024 - Bit Position > 35
25	You Have File Accessed Without Write Permission
26	You Have File Accessed Without Read Permission
27	Address Is Outside of File Limits
28	028 - Read Linked Files Only With This Command
29	029 - Illegal System Selection
30	134 - Invalid Function Number In DRL FILACT
31	135 - User Cannot Access the System Master Catalog
32	138 - TAP* File Undefined
33	139 - Error in Writing TAP* File
34	DRL ABORT - File Not Saved
35	DRL ABORT - Program Dumped
36	Not Enough Core to Run Job
37	Sorry - Out of Swap Space. Try again.
38	File Address Error - Retry
39	DRL ABORT - Error on Program Dump
40	DRL ABORT - ABRT File Too Small for Entire Dump.
41	Bad Status - DRL SAVE/RESTOR File
42	SAVE/RESTOR Filename Not in AFT
43	064 - Execute Time Limit Exceeded
44	Illegal Break Vector
45	065 - Object Program Size Limit Exceeded
46	Incorrect Entry to DRL TASK
47	RESTOR Program Name Undefined
48	Reached Catalog Limit on DRL SAVE
49	DRL PSEUDO: Tally Incorrect
50	Bad DRL SAVE Data Location
51	DRL SAVE - Do First Program Save Before Added Prog.
52	DRL SAVE On Random File Only
53	Terminal Type Not Allowed DRL PSEUDO

PRINT ERROR MESSAGE

PNTERR (EP1 of TSSG) prints a preset message onto a remote device and returns to LINSRV.

PRECALLING SEQUENCE

Prior to entering PNTERR, the X3 register must contain the error comment number.

CALLING SEQUENCE

PNTERR is called from DRLKON, DRLKOT, DRLDIO, TSXSPF, FILACT, CORFIL, USERID, CALLSS, DRLABT, RESTOR, EXENTR, ALLOC, ALL7AC, SWAP, SAVE, TASK, FAKEIN, SCAN1, SCAN2 and SCAN3 via:

```
8         16
TSX1      PNTERR
Return to LINSRV
```

For additional details, see PNTTSX.

SPECIAL ENTRY

PNTTSX (EP2 of TSSG) prints a preset message onto a remote device and returns to the caller.

PRECALLING SEQUENCE

Prior to entering PNTTSX, the X3 register must contain the error comment number.

CALLING SEQUENCE

PNTTSX is called from the DRLABT, RESTOR, STARTP, SAVE, EXENTR and ALL7AC via:

```
8       16
TSX1    PNTTSX
Return to caller
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

PNTTSX returns to the calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The procedure involves the use of the pseudo-courtesy call for remote I/O in the user status table for this user. As each terminate is recognized by the Line Service module, the courtesy call is honored and the next portion of the operation started. There are four phases of operation possible:

1. I/O in progress
2. Flush buffer
3. Print error message
4. Resume operation

If I/O is not in progress the first phase is skipped. If there is room in the buffer, the second and third operations may be combined.

The first two entries in the text portion of the table are XED instructions. These are pointers to entry points for the third and fourth phases of the sequence. The courtesy call is then set to point to the proper XED after the keyboard I/O is initiated. The code for each phase obtains the location of the XED which defines the proper text message. The indicated action is taken and return is made to Line Service.

On entry to the routine, the subsystem is taken out of Allocator's queue by a call to ALLOC with a kill code. When the last phase is complete, the process is resumed by asking the user to reselect a system. All files are preserved so the user may make corrections and resume his process.

Programming Method

PNTTSX is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)
System Message Generator SYS (EP3 of TSSI)
Buffer Dump BUFDMP (EP4 of TSSK)
System Error ERROR (EP1 of TSSF)
Transmit Data to Buffer STOBUF (EP6 of TSSK)
Set Tally for Output Buffer STBTAL (EP7 of TSSK)
Line Service LINSRV (EP1 of TSSJ)
Send "Stop Paper Tape" to DATANET PPTOFF (DRL STOPPT, TSSK)

CONTROL OF USER PROCESSES (TSSH)

The Control of User Processes subprogram (TSSH) controls the sequence of events occurring when the user selects a subsystem or inputs a command language word. This sequence is dependent on the subsystem or command language used.

This organization avoids the definition of one global set of command language for all subsystems and provides a reasonably flexible and powerful capability for defining logical sequences of actions at the time a subsystem is entered into the system. A time-sharing system in which the command language and the associated responses are defined within the executive would make it difficult to incorporate new subsystems into the time-sharing system with new command language words and associated actions.

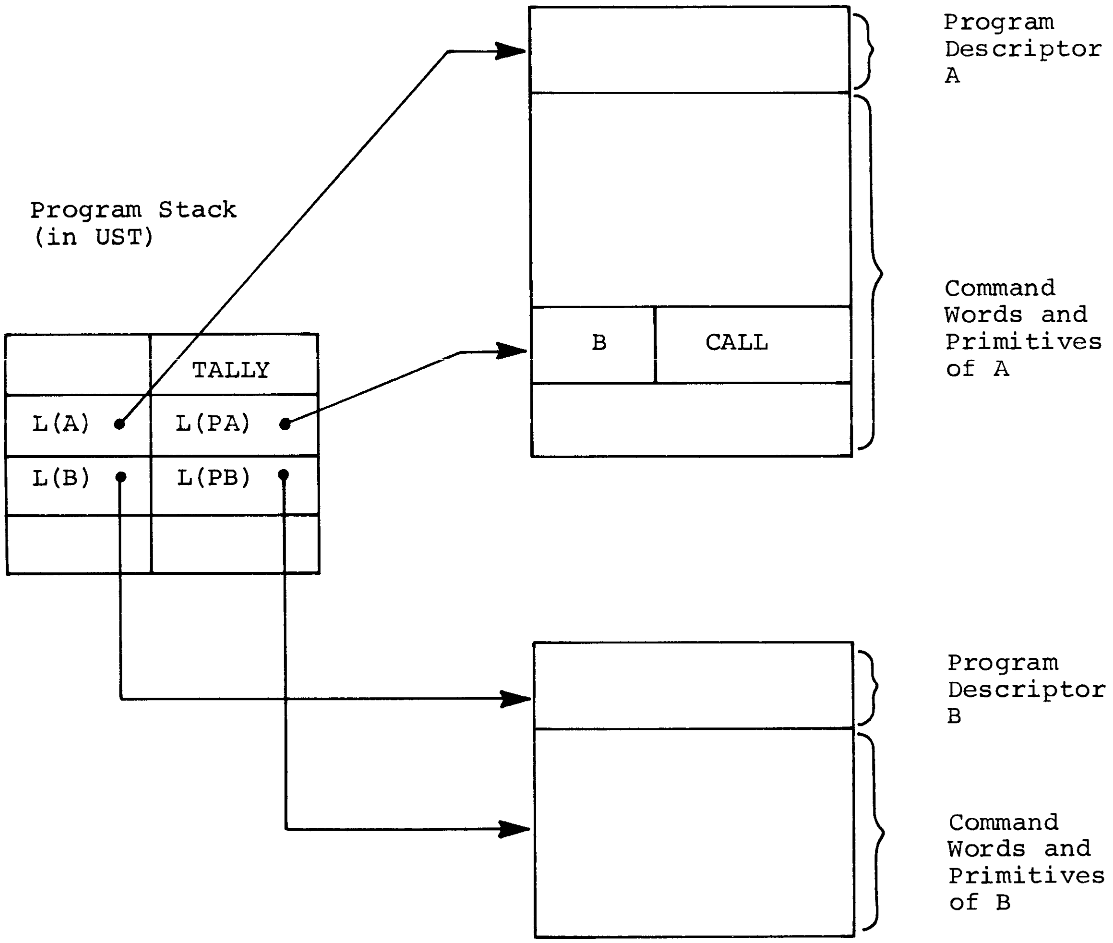
Each subsystem in the time-sharing system includes a program descriptor. A part of this program descriptor can be a list of command language to be recognized while the subsystem is active. Each of these defined command words has an associated list of primitives defining a sequence of actions to be performed by TSS. The primitives are high-level control functions which can call other subsystem programs, initiate an input stream from the remote terminal, etc. In addition to the sets of primitives associated with each command word, there is a set of startup primitives used for initial entry into the subsystem. In many cases, it is desirable to go through preliminary functions, such as an edit, before actually entering the selected subsystem program.

In this framework, each subsystem program is considered to be a self-contained element of the total process in that TSSH selects and loads the program (EXEC primitive) and the program returns to TSSH, which selects the next primitive and continues its flow of logic from that point. One of the primitives returns to the build mode in order to resume input from the keyboard and scan for more command language, providing a large amount of flexibility and control of the logical flow by the user.

TSSH uses the User Status Table associated with each user and the program descriptor tables to direct the flow for a given user. Pointers to the program descriptor area and the current primitive are kept in a list in the UST. Since it is possible to have several levels of control, this is a pushdown list (called program stack) with the last entry having the active pointers.

The program descriptor contains the command language and associated primitives for a given program. Included in the program descriptor is a startup sequence to be executed when the program is initially selected or called.

The following diagram shows the relationships between the pointers in the program stack and the program descriptors and primitives.



TSSH contains three entry points:

- EP1 Scan for Control Language (SCAN1)
- EP2 Scan for Known Command Word (SCAN2)
- EP3 Interpret Primitive (SCAN3)

SCAN1 and SCAN2 are used by Line Service to perform the scan for and initiation of processes based on command language. SCAN3 is used by STARTP and the Derail Processor to initiate execution of the log-on and log-off subsystems and to interpret the RETURN primitive.

TSSH occupies approximately 470 (octal) core storage locations.

Summary of Primitive Actions

- CALLP a (where a is program descriptor location)

  CALLP transfers control to the subsystem program descriptor located at a. TSSH finds the startup sequence from the program descriptor and takes its next primitive from this list. CALLP may occur in any list of primitives of a program descriptor and interrupts the execution of primitives from the present list. However, it is normally necessary for TSSH to return to that level after a series of functions performed by the called program are complete. The location of the primitive when the CALLP was encountered is saved in a pushdown list associated with the user status table. Thus, it is possible to have several levels of calls and to be able to resume operation at a previous level. (The primitive POPUP will resume operations at the previous level.)

- EXEC

  EXEC initiates loading and execution of the current subsystem program. This is accomplished by placing the job in the new interaction queue for Allocator by calling ALLOC with a code of 1. When the subsystem program has completed its functions, it returns to Executive via a derail operation and the next primitive in the sequence will be executed.

- BIN

BIN initiates the build mode of terminal input. While in this mode, Line Service reads and accumulates data on a collection file (SY**) for this user. Each line of input is scanned for the command language associated with this subsystem program. This scan is done by SCAN1, which is called by Line Service each time a line is read from the remote keyboard. If no command language is found, Line Service accumulates the input in buffer and dumps it when required to the user's collector file. If a command word is recognized, SCAN1 notifies Line Service which does the necessary housekeeping before the command is executed. A subsequent call to SCAN2 starts execution of the referenced primitives. Note that once the build input primitive is encountered, there is no "next" primitive implied. The next primitive is defined when a command word is encountered.

- POPUP

POPUP indicates that processing at this level is complete and processing at the previous level is to be resumed. SCAN3 reduces the tally, obtains the previous set of pointers from the User Status Table, obtains the next primitive and continues the flow of control from that point. However, if the subsystem was invoked by a DRL CALLSS from the previous subsystem, the caller of the subsystem is restored to the Allocator queue and the control goes to the Line Service. If the previous level does not exist (that is, this was the first level of control), POPUP calls the system routine which will ask the user which subsystem program he wishes to select next. All files defined during previous calls will remain defined.

- RETURN

RETURN advances the primitive pointer by one and resumes execution of primitives. It is not a true primitive in that it is not entered into the primitive list in program descriptors, but will be executed when a subsystem does a DRL RETURN to the Executive. The Derail processor enters SCAN3 which executes the RETURN primitive and resumes the execution at the next primitive.

- IFALSE n,e a

IFALSE provides for conditional execution of blocks of primitive operators. The conditional test is based on the user switch word (.LSWTH) in the User Status Table. The interpretation is, "If bit n is false (off), transfer control to the block of primitives at location a." If the test is true (on), control passes to the next primitive in sequence. This function allows considerable interaction between the execution of subsystem programs and the interpretation of primitives. A subsystem can also (via the specified derail) set or reset these switches.

- IFTRUE n,e a

Interpretation of IFTRUE is the same as IFALSE, except that transfer of control passes to a if bit n of .LSWTH is true (on).

- STRUE n,a

STRUE provides the capability of setting bits ON in the switch word (.LSWTH) in the individual User Status Table, for subsequent testing by IFALSE and IFTRUE. The setting and testing of these bits allows considerable interaction between subsystem programs, since they can pass information to each other by means of the switch word. Subsequent subsystem programs can then execute different blocks of code, based on the settings made by previous primitives, which could be different for different sequences of primitives. The interpretation of the above primitive is, "Set bit n true (ON) in the current User Status Table, and transfer control to the block of primitives starting at location a."

- STFALS n,a

The interpretation of STFALS is the same as STRUE, except that bit n is set false (OFF).

SCAN FOR CONTROL LANGUAGE

SCAN1 (EP1 of TSSH) searches for command language but takes no further action. (See SCAN3 for example.)

PRECALLING SEQUENCE

Prior to entering SCAN1, the registers listed must contain the data indicated.

X2	UST address
X3	Address of (possible) command word

CALLING SEQUENCE

SCAN1 is called from STG10 via:

```
 8         16
 TSX1      SCAN1
 Return    (Control language found)
 Return    (Control language not found)
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

SCAN1 returns to the calling routine.

POSTCALLING SEQUENCE

X3 is not saved

SUPPORTING INFORMATION

See SCAN3

Programming Method

SCAN1 is reentrant and relocatable.

Interrupts are inhibited while scanning.

Storage

No internal temporary storage is used.

Other Routines Used

Trace Recent Events TSEVNT (EP1 of TSSB)

SCAN FOR KNOWN COMMAND WORD

SCAN2 (EP2 of TSSH) initiates action based on the command language word found, then falls through to SCAN3. (See SCAN3 for example.)

PRECALLING SEQUENCE

Prior to entering SCAN2, the registers listed must contain the data indicated.

X2	UST address
X3	Address of command word

CALLING SEQUENCE

SCAN2 is called from STG10 via:

```
8          16
TSX1       SCAN2
Error return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

If command word found, falls through to SCAN3; otherwise, error return.

POSTCALLING SEQUENCE

A contains primitive to be interpreted.

SUPPORTING INFORMATION

See SCAN3.

Programming Method

SCAN2 is nonreentrant and relocatable.

Interrupts are inhibited while scanning.

Storage

No internal temporary storage is used.

Other Routines Used

None.

INTERPRET PRIMITIVE

SCAN3 (EP3 of TSSH) initiates execution of the log-on and log-off subsystems and interprets the RETURN primitive.

PRECALLING SEQUENCE

Prior to entering SCAN3, the registers listed must contain the data indicated.

X2	UST address
A	Primitive to be interpreted
X0	Present primitive

CALLING SEQUENCE

SCAN3 is called from STARTP and DRLRET via:

```
8          16
TSX1       SCAN3
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

SCAN3 returns to the calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Example of Flow

As an example of the flow through TSSH, a simplified set of program descriptors for subsystems associated with the BASIC compiler is defined--LIST and EDIT. The actions generated when a user goes through a normal sequence of inputting and running a new program will be as follows:

Program Descriptor for BASIC

Command Language for BASIC

LIST	Command Language Word
CALLP EDIT CALLP LIST BIN	Primitives for LIST
RUN	Command Language Word
CALLP EDIT EXEC BIN	Primitives for RUN
DONE	Command Language Word
POPUP	Primitive for DONE
(Startup) BIN	Primitive to be executed to initiate BASIC when called from other primitive or EXEC

Program Descriptor for EDIT

Command Language for EDIT

(None)	No Command Language
(Startup) EXEC POPUP	Startup Procedure for EDIT

Program Descriptor For LIST

Command Language for LIST

(None)	No Command LANGUAGE
(Startup) EXEC POPUP	Startup Procedure for LIST

User Actions

A sequence of user actions at the terminal is now defined.

1. Dial up, reply to log-on sequence, and then the time-sharing system requests SYSTEM ? . Response is BASIC.

2. The user types in a series of BASIC statements (not necessarily in order).

3. The user types in LIST.

4. The user types more statements.

5. The user types RUN.

6. The user types DONE.

At step 1, Line Service takes the results of the SYSTEM question and initiates execution by calling a control routine, STARTP. This simulates a CALLP BASIC primitive which then sets up the first level pointers in the User Status Table and proceeds to execute the startup sequence for BASIC.

This primitive is the BIN and will cause SCAN3 to set the proper flags and ask Line Service to initiate the accumulation of input data. The user now enters step 2 and types in source statements for BASIC. These are accumulated by Line Service into a file (SY**) for this user. The user enters step 3 by typing the command word LIST. LIST is detected by SCAN1 (via a call from Line Service). After Line Service flushes its buffers, SCAN2 starts execution of the sequence of primitives associated with LIST. The primitive pointer is set to the first primitive which is retrieved and executed. This is a CALLP EDIT and is required to sort the file before listing the output.

A new entry is made in the program stack in this User Status Table and the program descriptor entry for EDIT is found. The startup procedure for EDIT is then found and the current primitive pointer set to this location. The primitive is retrieved and interpreted as EXEC.

SCAN3 will request that Allocator load and execute the EDIT program. When EDIT has completed its task, it will perform a derail to the RETURN entry in Executive and SCAN2 will obtain its next primitive using the entry in the program stack at the previous level. This is CALLP LIST. Therefore, when the LIST program descriptor is found, a new entry is made in the program stack, and the same procedure as was performed for EDIT is again performed. That is, LIST is executed and the POPUP performed to return control at the previous level. The primitive here is now BIN. Input is initiated as before.

The user is now at step 4 in the sequence; the new input is being accumulated on his scratch file by Line Service, as before. At step 5, the user types the command word RUN. SCAN1 finds a command and then SCAN2 will find the associated string of primitives (both called by LS).

The sequence will be the same as before. A new entry is made in the program stack and the startup sequence is found and initiated. When the EDIT program returns, the POPUP primitive is encountered returning control to the next primitive in the RUN sequence (EXEC). SCAN3 makes this request of Allocator. When BASIC is loaded and in execution, BASIC compiles the source file data (now properly edited) and gives control to the compiled program. Eventually, it returns and SCAN2 finds the next primitive, BIN. The user can now make more corrections and repeat the sequence as often as is desired. When he is finished with BASIC, the user types DONE. This is a command language word and will be interpreted by SCAN2. This says POPUP, but there is no previous level, so SCAN3 asks Line Service to initiate the SYSTEM ? sequence and wait for the user to ask for another system. The user could, at this point, sign off or select some other system.

This example has been considerably simplified to facilitate the description of the flow. For example, no provision was made for using an old file or saving the source file, or for testing and setting switch word bits. Extensions to the command language make this possible.

See the more detailed example of flow in the programmer's reference manual, CPB-1514.

Programming Method

SCAN3 is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)
System Message Generator SYS (EP3 of TSSI)
Print Error Message PNTERR (EP1 of TSSG)
TSS Subsystem Startup STARTP (EP4 of TSSI)
Trace Recent Events TSEVNT (EP1 of TSSB)
UST Collect CUST (EP2 of TSSN)

UTILITY ROUTINES (TSSI)

The Utility Routines (TSSI) perform the startup, termination and accounting procedures required by TSS.

It contains six entry points:

- EP1 Terminate Routine (TERM)
- EP2 Clear UST and Make Accounting Entry (TRMCU)
- EP3 Initiate Remote I/O (KEYIO)
- EP4 System Message Generator (SYS)
- EP5 TSS Subsystem Startup (STARTP)
- EP6 Input Collector File Space Manager (SYMAN)

TSSI occupies approximately 730 (octal) core storage locations.

TERMINATE ROUTINE

TERM (EP1 of TSSI) clears the user's open-file list (releases temporary files and deaccesses permanent files).

PRECALLING SEQUENCE

Prior to entering TERM, the registers listed must contain the data indicated.

X2   UST address

CALLING SEQUENCE

TERM is called from TERMPG and NEWUSER via:

```
8         16
TSX1      TERM
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to LINSRV to pass on to next active UST.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

TERM is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Trace Recent Events TSEVNT (EP1 of TSSB)
File Deallocation PATDEA (EP2 of TSSD)
Line Service LINSRV (EP1 of TSSJ)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Delinking a File FREE (EP3 of TSSD)
Update Allocator Queue ALLOC (EP1 of TSSL)
End Courtesy Call ENCC (EP5 of .MDISP)

CLEAR UST AND MAKE ACCOUNTING ENTRY

TRMCU (EP2 of TSSI) constructs and writes a record on the accounting file, then clears and initializes the UST for the next user.

PRECALLING SEQUENCE

Prior to entering TRMCU, the registers listed must contain the data indicated.

```
X2      UST address
```

CALLING SEQUENCE

TRMCU is called from TERMPG and NEWUSER via:

```
 8         16
 TSX4      TRMCU
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

TRMCU returns to the calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

TRMCU is nonreentrant and relocatable.

Interrupts are inhibited throughout.

Storage

No internal temporary storage is used.

Other Routines Used

Trace Recent Events TSEVNT (EP1 of TSSB)
Update Allocator Queue ALLOC (EP1 of TSSL)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Accounting File Request ACTFL (EP13 of .MIOS)
MME GEINOS Processor INOS (EP5 of .MIOS)

INITIATE REMOTE I/O

KEYIO (EP3 of TSSI) initiates keyboard I/O for a particular user. Depending upon the contents of the A-register, a MME GEROUT output, output-then-input, or prepare for paper tape input will be issued by KEYIO. Upon calling KEYIO, the following requirements are assumed fulfilled:

1. The user is not presently engaged in remote I/O processing.

2. The direct build mode flag is correctly set for this user.

3. The current line pointer in the user's line buffer is pointing to the output message.

4. All message data is in the correct format within the user's line buffer (that is, word count, character count, input buffer address, etc.).

PRECALLING SEQUENCE

Prior to entering KEYIO, the registers listed must contain the data indicated.

```
X2      UST address
A      =0, output requested
       =1, output/input requested
       =-1, prepare for paper tape input
```

CALLING SEQUENCE

KEYIO is called from ERROR and BUFDMP via:

```
 8        16
 TSX1     KEYIO
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Normal return is to calling module.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

KEYIO is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Trace Recent Events TSEVNT (EP1 of TSSB)
Monitor Procedures MONCK (EP2 of TSSJ)
Conditional MME GEROUT Entry FGROUT (EP16 of .MFALT)
MME GEROUT Processor GROUT (EP1 of .MROUT)

SYSTEM MESSAGE GENERATOR

SYS (EP4 of TSSI) generates the terminal message, SYSTEM?.

PRECALLING SEQUENCE

Prior to entering SYS, the registers listed must contain the data indicated.

```
     X2      UST address
```

CALLING SEQUENCE

SYS is called from SCAN3, GOOD10, STGIO, SYSRET, and NEWUSER via:

```
 8        16
|         |
|TSX1     |SYS
|Return   |
|         |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to calling module.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

SYS initializes the user's buffer area and the program-stack tally word. A pseudo-courtesy call is inserted in the User Status Table, and the remote I/O flag and the CC-REQ'D flag are turned on. The combination of these two flags will result in Line Service executing the courtesy call when the terminate from the output is received. The message SYSTEM? is inserted in the buffer and the message sent to the terminal via GEROUT. The status return from the MME GEROUT is located in the user Status Table for this user.

When the pseudo-courtesy call is honored at the termination of the response from the user, the routine calls STARTP, to initiate operation of the system.

Programming Method

SYS is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Trace Recent Events TSEVNT (EP1 of TSSB)
Line Service LINSRV (EP1 of TSSJ)
Monitor Procedures MONCK (EP2 of TSSJ)
MME GEROUT Processor GROUT (EP1 of .MROUT)

TSS SUBSYSTEM STARTUP

STARTP (EP5 of TSSI) initiates the allocation of and dispatching to the requested subsystem.

PRECALLING SEQUENCE

Prior to entering STARTP, the registers listed must contain the data indicated.

X3	Pointer to ASCII name of subsystem
X2	UST address

CALLING SEQUENCE

STARTP is called from SCAN3, GOOD10, STGIO, CALLSS and NEWUSER via:

```
8         16
TSX1      STARTP
Return    (Program initiated)
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

STARTP obtains the word specified by X3 and scans for a carriage return or end of word. The result is a name left-justified and blank-filled. Using this name, the list of program descriptors is searched for an identical name. If a match is found, a CALLP primitive is constructed in the accumulator and an entry made to SCAN3. The routine then returns to the caller.

If the name is not found in the program descriptor list, control passes to the error printout routine PNTTSX which prints the message SYSTEM UNKNOWN and asks the user to reselect a subsystem.

Programming Method

STARTP is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Interpret Primitive SCAN3 (EP3 of TSSH)
Special Entry PNTTSX (EP2 of TSSG)

INPUT COLLECTOR FILE SPACE MANAGER

SYMAN (EP6 of TSSI) allocates and deallocates input collector files (SY**) in increments of 320-word blocks out of the file space acquired by TSTART.

PRECALLING SEQUENCE

Prior to entering SYMAN, the registers listed must contain the data indicated.

```
X2      UST address
A = 0,  Allocate an SY** for the UST
  ≠ 0,  Deallocate an SY** for the UST
```

CALLING SEQUENCE

SYMAN is called from DYUST, RUST and NEWU76 via:

```
8         16
TSX1      SYMAN
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

This routine uses a table, .TFBLK, set by Startup (TSTART). The table contains enough link descriptions to provide an SY** for each user. The links are partitioned into blocks and the PAT built on allocation. A bit in .TFBLK is set to mark the block used. An indication of the link and block used is kept in the PAT pointer. PATMAN is called to place the SY** file in the available File Table.

On deallocation, the PAT is released and the bit in .TFBLK reset to indicate that the block is again available.

Programming Method

SYMAN is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Delinking a File FREE (EP3 of TSSD)
File Entry Space Allocation PATMAN (EP1 of TSSD)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

LINE SERVICE (TSSJ)

Line Service (TSSJ) provides the interface between the time-sharing system and the remote terminal system (GERTS) portion of GECOS III. (A remote-terminal connection with TSS is handled by GERTS in essentially the same manner as for any other slave program.) The line-service interface performs -- or is involved in -- the following functions:

1. Entering new users into the time-sharing system.
2. Processing all remote I/O terminations.
3. Accumulating build-mode input for each user.
4. Enabling build-mode command-language scanning and processing.
5. Periodic status reports to the main system console.

Line Service initiates most remote, or keyboard, I/O except direct-mode, subsystem-requested I/O (via DRL's KOUT, KOUTN, and TAPEIN) and handles all remote-I/O terminations. Build-mode remote I/O, and the file I/O associated with it, is initiated exclusively within Line Service.

Line Service is divided into several main portions, corresponding to the functions listed above. The passage of processor control through the different portions of the program corresponding to (1) and (2) is time-dependent; that is, each section gains control only if the duration of time since its last execution exceeds a specified interval, t, where the interval chosen is directly related to the priority of the function and the time required for its execution. (The time intervals will be periodically redefined, if required, as dictated by performance studies to provide the most efficient system flow.) Processor time spent in build-mode processing (3 and 4, above) is primarily controlled by MME GEROUT courtesy calls. (Initiation of build-mode for each user is enabled by TSSH, via flag settings for Line Service.)

TSSJ contains two entry points:

- EP1 Line Service (LINSRV)
- EP2 Monitor Procedures (MONCK)

TSSJ also contains five return points from TSSJ extensions in TSSO.

- System Loaded-Reject New User (LDED)
- Enter new User in UST and Update NUQ Loop (GOOD10)
- Update to Next Entry (UPDAT)
- Build Mode Processing of Input (STGIO)
- Paper Tape Courtesy Call (PPTCC1)

TSSJ occupies approximately 2530 (octal) core storage locations.

Entering New Users Into Time-Sharing

Following time-sharing startup, Line Service issues a MME GEROUT (remote inquiry) with the program name "GE-TSS." The location of this MME and the program name remains in GECOS III direct-access queue until a keyboard user requests a direct-access connection to this program. At this time, GECOS III inserts the user's 2-character station identification into the MME sequence and deletes this entry from its queue.

Periodically, Line Service examines this MME for a nonzero station-ID field. When such a condition is encountered, Line Service places this station ID in TSS's new user queue and immediately reinitiates the reinitialized MME GEROUT, so that a remote inquiry is outstanding at all times.

It is possible that more than one user has requested connection to the time-sharing system in the time interval it takes Line Service to process the new station ID and initiate another MME GEROUT. Therefore, Line Service cycles this MME until a zero station ID is present, queuing the new ID's in the new user queue.

When no more new users are present or the new user queue is full, Line Service processes the entries in the new user queue. Assuming that the time-sharing system is capable of handling m users simultaneously, Line Service tests the number of users presently on the system. If the system is loaded, LDED issues a system busy or loaded message to the prospective user, automatically disconnects the line, and deletes the entry from the new user queue. If the system is capable of accommodating another user, Line Service increments the count of users on the system and enters the station ID in the first available User Status Table (or calls DYUST). GOOD10 initializes accounting and statistics for the user, and enables the log-on sequence for this user via a TSX1 STARTP to call the NEW subsystem. Subsequently, SYS is called.

When no users are present on the time-sharing system, Line Service sets an alarm clock and zero urgency. This allows time-sharing to be swapped out. When a terminal user requests the time-sharing system, GECOS III swaps time-sharing in, enters the user identification into the outstanding remote-inquiry entry, and enables the time-sharing system.

Processing Keyboard I/O Terminations

The main portion of Line Service processes keyboard termination for both completed input and output requests. This section of Line Service is processed every n milliseconds.

The amount of time spent in terminate processing is dependent upon the number of terminates, the number of buffer dumps to the disc/drum, and the normal/abnormal I/O terminate processing.

Line Service begins terminate scanning at the first UST entry. To be eligible for Line-Service processing, the entry must satisfy the following tests:

1. Remote-I/O-in-process bit is on.

2. GEROUT has placed a nonzero keyboard-I/O status (or the executive, a minus 1) in the entry's remote status word, .LTTYS.

(If the I/O status word is a minus one, Line Service assumes this to be the termination of some operation (e.g., line-buffer-flushing file I/O, implying a continuation of build-mode for that user.)

If the entry is eligible for terminate processing, Line Service examines the I/O completed status for abnormalities, in which case Line Service enables a recovery routine or indicates, in the UST, that I/O transmission was abnormal.

If the keyboard I/O terminated with a normal status, Line Service determines if an all-points bulletin (APB) is to be sent to this user and, if so, outputs the message from the .TCCMT location in the time-sharing system's Communication Region. All flags are saved so that upon completion of the APB, control will be passed as it would have been before it was issued. Line Service then resets the remote I/O in process bit and then tests for CC REQD bit on. If the CC REQD bit was on, Line Service enables the required processing by turning off the CC REQD bit and transferring control to the entry's courtesy call word, which was preset by some portion of the executive (possibly Line Service itself). Upon return from the pseudo-courtesy call, Line Service tests whether the terminate cycle was completed. If not, it proceeds to the next active User Status Table via UPDAT.

This procedure for passing control to a block of code at terminate time via the CC REQD bit is analogous to the true courtesy call for GECOS III I/O. The pseudo-courtesy call feature is used throughout the time-sharing system executive to initiate remote I/O and gain control at terminate time for updating status, initiating other processes, etc. The necessary procedure is:

1. Set a transfer in user's courtesy call address, .LCC, to the block of code to be executed on a remote terminate (normally TSX3 --).

2. Set remote-I/O flag on and set CC REQD bit on.

3. Initiate I/O.

When the terminate status is found and the CC REQD bit is also on, Line Service does a TSX1 to the user's courtesy call entrance via .LCC. When the required function is completed, a return to the next Line Service instruction is expected (that is, TRA 0,1).

If the CC REQD bit was not set (and assuming non-build-mode I/O), Line Service has completed all necessary terminate processing (that is, reset the remote I/O in process bit for the entry), and returns to the next UST entry or exits.

The terminate processing performed cyclicly every n milliseconds primarily handles direct-mode (i.e., subsystem-initiated) and system-level I/O, plus abnormal build-mode termination processing and some returns from file I/O issued by Line Service. The bulk of build-mode terminate processing (described below) is performed within MME GEROUT courtesy calls, in order to speed up this function, but both modes of terminate processing are performed by the same code, virtually all of which is common to both.

Direct-mode and executive-initiated remote I/O is issued without a (true) courtesy call.

Accumulating Build Mode Input

Upon receipt of a MME GEROUT courtesy call, the status word (.LTTYS) is checked for abnormal status (break, disconnect, etc.) and if so, the courtesy call is ended, with the remote-I/O-in-process bit left on to allow this status to be dealt with during the next cyclic terminate processing.

If the status was normal, the CC entrance bit is set. If the user is in auto mode and the current line pointer (.LSTEM+1) does not exceed the bounds of the line buffer threshold value, then either STGIO issues a line feed message and a new line number or changes the user to normal mode (i.e., not auto mode) and issues a line feed message and asterisk. Which action is taken depends on whether or not the user entered more than a CR (carriage return) in the previous line. If only a CR was issued, the second action is taken. If the user is not in auto mode or if the current line pointer exceeds the bounds of the line buffer, then STGIO does a TSX1 to SCAN1 in TSSH to determine whether or not the last line contained a command (previous line pointer in .LSCLP,2). If the return from SCAN1 indicates no command language in the input line, the current line pointer value for the user's line buffer is tested.

If the current line pointer exceeds the bounds of the line-buffer threshold value, appropriate flags are set and the courtesy call ended. On the next cycle, Line Service locates the user's collector file from his file table, and issues a MME GEINOS, with a courtesy call, to write the loaded buffer onto the disc. A TSX3 RTNCM is inserted in the .LCC location of the user's UST, and the MME courtesy call address is modified to point to the indicate that the file I/O was initiated and its termination must be handled by Line Service. Upon file termination, the courtesy call directs control to the user's .LCC location and executes the TSX3 RTNCM (the contents of XR3 indicate which UST is being processed). RTNCM, in Line Service, interrogates the file terminate status, and if normal, updates the current line pointer to the top of the line buffer. It resets the line buffer and sets the I/O status word to minus one with the remote-I/O-flag on (to indicate Line Service should continue build-mode input during the next processing cycle) and ends the courtesy call.

If the current line pointer was within bounds, a line-feed and asterisk message is sent to the user during the original GEROUT courtesy call, to allow the input of an additional line from his terminal.

In the event that an abnormal status is detected at the end of a MME GEROUT courtesy call, TSSJ tests that status to determine which of three possible situations exist - a remote disconnect, a BREAK received from the remote terminal, or a GBUSY, which indicates a timing error internal to the communications subsystem, in which a previous function did not properly terminate before the next function started.

If the abnormal status is a GBUSY, control is transferred to ERROR (EP1 of TSSF). Control then passes to UPDAT in TSSJ, which processes the next user.

If the disconnect code is received, TSSJ checks with MONCK to determine if monitoring is in progress, and whether or not it is the master or monitored user who is disconnecting. If any of these conditions are true, monitoring is discontinued and the user in question is singled out. Termination then continues normally, with a call to ALLOC (EP1 of TSSL) with a code 1 to inform the allocator of the disconnect. Following wrapup accounting, TSSJ calls STARTP (EP5 of TSSI) and when TSSI returns, control is passed to UPDAT.

If the code received indicates a "BREAK", TSSJ checks to see if the user is already terminating, and if so, ignores the code and returns to UPDAT. If not, checks for monitored and/or master users, or 760 terminals are made, and preliminary exception processing is done for them. Eventually, all I/O is reset, and a check is made to determine whether or not the user currently has a subsystem allocated.

If no subsystem is in use, TSSJ either pops up to the next primitive or returns to SYSTEM level, depending upon the condition of the user's stack, and exits to UPDAT.

If a subsystem is in use, TSSJ determines whether or not it wants BREAKS passed back to it. If so, bit 35 in .LFLAG is set ON to warn TSSM, and control returns to UPDAT. Otherwise, code 1 is passed to ALLOC (EP1 of TSSL), and control returns to UPDAT.

Enabling Command Processing In Build Mode

If return from SCAN1 indicates that the input line contained a command, STGIO sets appropriate flags, and ends the courtesy call. On the next processing cycle, an EOF indicator is inserted in the upper half of the character count word of the command message and the user's preceding data lines (located in the buffer) are appended onto the user's collector file prior to enabling the command processing. A disc/drum write with courtesy call is issued, with return to RTCMD upon disc/drum termination. (The method of issuing the courtesy call is the same as that discussed in the paragraph above with the exception that an RTCMD is used instead of RTNCM.) Flags FILE I/O IN PROCESS and LS FILE I/O are set on to indicate that the file I/O was initiated and its termination must be handled by Line Service.

When the courtesy call is honored, RTCMD interrogates the file terminate status, and if normal, places a TSX3 DOCMD in the UST entry's .LOC location, sets appropriate flags, and ends the GENIOS courtesy call. Eventual execution of the TSX1 DOCMD causes a call to SCAN2, in TSSH, to initiate processes based on the command word. This action ends the build-mode cycle for that user.

Termination

On a given entry to the Line Service terminate processor, all lines with I/O in process are checked for terminates. If no terminates were received and time-sharing has nothing to do until a terminate is received, the Line Service routine sets an alarm clock with normal urgency. This allows batch jobs to use the processor with no interference from time-sharing, but does not normally allow time-sharing to be swapped. A terminate received from any I/O or a new user connection will break the alarm clock and GECOS III reenables time-sharing.

Keyboard input and output data is temporarily collected in line buffers. The line buffers are permanently assigned to the UST. At initialization time, each UST is initialized with the address of the line buffer which is a part of the UST.

Note that on termination of a file I/O operation, the remote-I/O-in-process bit is often set in conjunction with other bits or a special flag in the status word, in order to simulate a remote I/O termination for the next cycle of terminate processing. Also, a bit in the second flag word, coding due to a GEROUT courtesy call, and an entrance to the same coding due to cyclic execution of the terminate processor (CC-ENTRANCE bit). Since much of the same code is executed at main level and at courtesy-call level, the distinction is necessary at these points:

1. Issuance of MME GEROUT (separate code exists for calls at either level).

2. Termination of the coding sequence, i.e., MME GEENDC or continuation within Line Service.

LINE SERVICE

LINSRV (EP1 of TSSJ) attempts to perform the following functions in the order listed:

1. Write console messages.
2. Check for new users on line.
3. Process new users in NUQ.
4. Check for terminals not doing remote I/O and process status for each.
5. Check for terminals doing remote I/O and process status for each.

The successful performance of each operation is dependent on timer limitations.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

LINSRV is called from EXENTR, ALLOC, ALLOCI, ALL7AC, SWAP, NEWU76, SYS, TERMPG, CALLSS, NEWUSER, SYSRET, DRLABT, RETSBS, DRLRFL, PASFIL, RESTOR, GROW, CONSOL, PNTERR, DRLRET, TAPEIN, DRLDIO, FILACT, STPSYS, PPTOFF, PPTERM, TASK, and RELMEM. Control is never returned to the calling routine.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

TRA to WRAPUP when aborting TSS; to ALLOCI when users are on the system but not using LINSRV; ZERO URGENCY when no users on the system. LINSRV makes TSS available for swap.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

LINSRV is reentrant and relocatable.

Interrupts are inhibited while checking for new ID.

Storage

No internal temporary storage is used.

Other Routines Used

Provide Date and Time of Day FGTIM (EP8 of .MFALT)
MME GEINOS Processor INOS (EP5 of .MIOS)
Release #S Links RELSPC (EP3 of TSSC)
End Courtesy Call ENCC (EP5 of .MDISP)
Core Size Adjustment ALL7AC (EP3 of TSSL)
Line Service Check EXTSQ (EP2 of TSSM)
Trace Recent Events TSEVNT (EP1 of TSSB)
Dynamic UST Generator DYUST (EP1 of TSSN)
MME GEROUT Processor GROUT (EP1 of .MROUT)
New DATANET-760 User GOOD76 (EP2 of TSSO)
TSS Subsystem Startup STARTP (EP5 of TSSI)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Relinquish Control Until Program Enabled DSCNT (EP11 of .MDISP)
Set Alarm SCK (EP13 of .MDISP)

Allocate Job in Queue ALLOCI (EP2 of TSSL)
Monitor Procedures MONCK (EP2 of TSSJ)
Update Allocator Queue ALLOC (EP1 of TSSL)
Interpret Primitive SCAN3 (EP3 of TSSH)
System Message Generator SYS (EP4 of TSSI)
System Error ERROR (EP1 of TSSF)
Scan for Control Language SCAN1 (EP1 of TSSH)
Executive Disc I/O TSXDIO (EP3 of TSSK)
Scan for Known Command Word SCAN2 (EP2 of TSSH)
Special Entry PNTTSX (EP2 of TSSG)

MONITOR PROCEDURES

MONCK (EP2 of TSSJ) allows a master user to monitor the interaction between another user and the system.

PRECALLING SEQUENCE

Prior to entering MONCK, the registers listed must contain the data indicated.

X2	UST address

CALLING SEQUENCE

MONCK is called from the SYS, ALLOCI and KEYIO routines via:

```
8        16
TSX1     MONCK
Return   master user
Return   monitored user
Return   no monitoring
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to call+1 if unsuccessful.
Return is to call+2 if successful.

POSTCALLING SEQUENCE

X7 Contains address of master UST if either master or monitored user is found.

SUPPORTING INFORMATION

Programming Method

MONCK is reentrant and relocatable.

Interrupts are inhibited while checking for master or monitored user.

Storage

.MTEMP is used to store X2.

Other Routines Used

MME GEROUT Processor GROUT (EP1 of .MROUT)
System Message Generator SYS (EP4 of TSSI)

SYSTEM LOADED - REJECT NEW USER

The LDED function of TSSJ issues a busy message to new user and sets flags for a disconnect when a new user is in NUQ but the system is loaded. This is also a return point from LINSRV extensions in TSSO.

Prior to entering LDED, the registers listed must contain the data indicated.

X5	Station ID
X7	NUQ entry index
A	NUQ entry word

GOOD10 GOOD10
TSSJ TSSJ

ENTER NEW USER IN UST AND UPDATE NUQ LOOP

The GOOD10 function of TSSJ updates the count of users, enables the new user, sets flags for APB and continues to process the next new user. This is also a return from LINSRV extensions in TSSO.

Prior to entering GOOD10, X2 contains the UST address.

UPDATE TO NEXT ENTRY

The UPDAT function of TSSJ finds the next terminal with its remote I/O flag on while LINSRV is processing the status of such terminals. This is also a return point from LINSRV extensions in TSSO.

Prior to entering UPDAT, X2 contains the UST address.

BUILD MODE PROCESSING OF INPUT

The STGIO function of TSSJ initializes pointers for input, checks line mode, tests for a command input and starts appropriate action. This is also a return point from LINSRV extensions in TSSO.

Prior to entering STGIO, the registers listed must contain the data indicated.

X2	UST address
X3	Current line pointer

PAPER TAPE COURTESY CALL

The PPTCCl function of TSSJ processes paper tape input from the terminals and places the input on the TAP* file.

Prior to entering PPTCCl, X2 contains the UST address.

Entry is via a pseudo-courtesy call from DRLKON.

DERAIL PROCESSORS (TSSK)

The Derail Processors satisfy requests for executive services by a subsystem program in execution. A derail (DRL) instruction causes a fault to be returned via GECOS III to the TSS fault vector. The TSS executive modules then perform the requested action and return to the calling subsystem or another subsystem as required to process the request.

The derail functions are similar to and replace MME functions in the GECOS system since a subsystem may not execute a MME instruction. The address portion of the DRL instruction defines the function to be performed. (See the Transfer Vector Table which follows.)

In addition to the DRL routines, the TSSK module also contains nine entry points as follows:

- EP1 Derail Entry from Executive (EXDRL)
- EP2 Return to Subsystem from Derail (RETSBS)
- EP3 Executive Disc I/O (TSXDIO)
- EP4 Buffer Dump (BUFDMP)
- EP5 Set Tally for Output Buffer (STBTAL)
- EP6 Transmit Data to Buffer (STOBUF)
- EP7 Executive File Spacing (TSXSPF)
- EP8 Search the AFT (SRCH)
- EP9 Switch File Names (TSXSWH)

TSSK occupies approximately 6130 (octal) core storage locations.

Transfer Vector Table

	Transfer to		Decimal	
DRL	Derail Processor		Address	Function
DIO	TRA	DRLDIO	1	DISC IO
KOUT	TRA	DRLKOT	2	KEYBOARD OUTPUT
KOUTN	TRA	DRLKON	3	KEYBOARD OUTPUT/INPUT
KIN	TRA	DRLKIN	4	KEYBOARD INPUT CURRENT LINE
RETURN	TRA	DRLRET	5	RETURN FROM SUBSYSTEM
DEFIL	TRA	DRLFIL	6	DEFINE A FILE
ABORT	TRA	DRLABT	7	ABORT JOB
SETSWH	TRA	DRLSET	8	SET SWITCH WORD
RSTSWH	TRA	DRLRST	9	RESET SWITCH WORD
REW	TRA	DRLREW	10	REWIND FILE
FILSP	TRA	DRLSPF	11	SPACE A FILE(BACK-FWD SPACE)
RETFIL	TRA	DRLRFL	12	RETURN A FILE

```
RELMEM     TRA    RELMEM      13    RELEASE MEMORY
ADDMEM     TRA    ADDMEM      14    ADD MORE MEMORY
CORFIL     TRA    CORFIL      15    DATA FROM/TO CORE FILE
SNUMB      TRA    SNUMB       16    OBTAIN SNUMB
TIME       TRA    TIME        17    OBTAIN TIME, DATE, PROCESSOR TIME
PASAFT     TRA    PASAFT      18    PASS AFT TO A SUBSYSTEM
TERMTP     TRA    TERMTP      19    OBTAIN TERMINAL TYPE
PDIO       TRA    PDIO        20    PRIVILEGED DISC I/O
RESTOR     TRA    RESTOR      21    RESTORE (OVERLAY LOAD)
SPAWN      TRA    PASFIL      22    PASS FILE TO BATCH
TAPEIN     TRA    DRLPPT      23    PAPER TAPE INPUT
CALLSS     TRA    CALLSS      24    INTERNAL CALL TO ANOTHER SS
USERID     TRA    USERID      25    GIVE EXEC USER ID ADD PRIORITY
TERM       TRA    TERMPG      26    USER TERMINATED-CLEAN UP UST
PASUST     TRA    PASUST      27    PASS UST TO SYBSYSTEM
MORLNK     TRA    MORLNK      28    REQUEST MORE LINKS
NEWUSR     TRA    NEWUSR      29    LOGON A NEW USER W/O DISCONNECT
FILACT     TRA    FILACT      30    PERM FILE REQUEST
SETLNO     TRA    SETLNO      31    SET LINE NO./INC. IN UST
SYSRET     TRA    SYSRET      32    RETURN TO SYSTEM LEVEL
STPSYS     TRA    STPSYS      33    FORGET SUBSYSTEM
STATUS     TRA    STATUS      34    CHECK I/O STATUS (Not Implemented)
DRLDSC     TRA    DRLDSC      35    DISCONNECT TERMINAL
PASDES     TRA    PASDES      36    PASS FILE TABLE AND DESCRIPTION
JSTS       TRA    JSTS        37    JOB STATUS
CGROUT     TRA    CGROUT      38    PROCESS LINE SWITCH
PART       TRA    PART        39    PARTIAL TEMP FILE RELEASE
GROW       TRA    GROW        40    GROW A PERM FILE
ABTJOB     TRA    ABTJOB      41    ABORT JOB
CONSOL     TRA    CONSOL      42    TALK TO 600 CONSOLE
SWITCH     TRA    DRLSWH      43    SWITCH FILE NAMES
DRLIMT     TRA    DRLIMT      44    STORE PROCESSOR TIME LIMIT
JOUT       TRA    JOUT        45    BATCH OUTPUT REQUEST
KOTNOW     TRA    OUTNOW      46    KEYBOARD OUTPUT FROM UNFILLED BUF.
OBJTIM     TRA    OBJTIM      47    PROC. TIME AND CORE SIZE LIMIT
PASFLR     TRA    PASFLR      48    PASS FILE TO REMOTE BATCH PROC.
STOPPT     TRA    PPTERM      49    SEND "STOP PAPER TAPE" TO DATANET
DRLSAV     TRA    SAVE        50    SAVE PROGRAMS ON PERM FILE
TASK       TRA    TASK        51    SPAWN A SPECIAL BATCH ACTIVITY
PSEUDO     TRA    FAKEIN      52    SIMULATED KEYBOARD INPUT
```

ENTRY FROM EXECUTIVE

EXDRL (EP1 of TSSK) stores the IC and I address in user's data region, finds the proper derail in the transfer vector table, sets X3 to point to the DRL address, sets X2 to point to the appropriate UST, and transfers control to the derail routine.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

EXDRL is called from EXENTR as a result of a DRL fault occurring within an active subsystem.

```
8       16
TSX1    EXDRL
Error return
Normal return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Normal return is to the specified derail processor. Error return is to DRLER1 to print DRL code.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

EXDRL is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Return to Subsystem from Derail RETSBS (EP2 of TSSK)

RETURN TO SUBSYSTEM FROM DERAIL

RETSBS (EP2 of TSSK) counts the number of consecutive returns to subsystem execution (limit, 30). If not over limit, returns to subsystem execution. If over limit, forces a return through LINSRV.

PRECALLING SEQUENCE

Prior to entering RETSBS, the X2 register must contain the UST address.

CALLING SEQUENCE

RETSBS is called from ALLOCI, EXDRL, TASK, DRLKON, DRLKOT, DRLKIN, DRLSET, DRLDIO, DRLSPF, DRLFIL, DRLRFL, PASAFT, CORFIL, SNUMB, TIME, TERMTP, USERID, TERMPG, NEWUSR, PASUST, MORLNK, PASFIL, RELMEM, ADDMEM, SETLNO, PASDES, JSTS, DRLSWH, CCROUT, PART, ABTJOB, PPTERM, and DRLIMT via:

```
 8       16
 TSX1    RETSBS
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

If the count of returns to subsystem execution is not over 30, return is to RETSSX in TSSM; if over 30, return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RETSBS is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

EXECUTIVE DISC I/O

TSXDIO (EP3 of TSSK) performs the I/O for routines within the TSS Executive. It sets flags, stores pointers, courtesy calls and status, and goes to DRLDIO for actual input/output.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TSXDIO is called from PASFIL, EXENTR, DRLABT, RESTOR, FLUSH, SAVE, TASK and ERRORT via:

```
 8         16
 TSX1      TSXDIO
 SEEK
 ZERO      L(file I/D),L(DCW1)
 READ/WRITE COMMAND
 ZERO      L(file I/D),L(DCW2)
 ZERO      L(status),L(courtesy call)
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to DRLDIO.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

TSXDIO is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

Internal temporary storage is used to save registers.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)

BUFFER DUMP

BUFDMP (EP4 of TSSK) dumps the buffer contents to the remote terminal.

PRECALLING SEQUENCE

Prior to entering BUFDMP, the registers listed must contain the data indicated.

```
     A     L(courtesy call),(0 or 1)
                             0 = Write only
                             1 = Write then read
```

CALLING SEQUENCE

BUFDMP is called from DRLKON, NEWU76, PNTERR, DRLRET, DRLKOT, FAKEIN, DRLDSC and SYSRET modules.

```
 8        16
|         |
|TSX1     |BUFDMP
|Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

BUFDMP is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Initiate Remote I/O KEYIO (EP2 of TSSI)

SET TALLY FOR OUTPUT BUFFER

STBTAL (EP5 of TSSK) sets the tally for the output buffer if it is not already set. If it is set, STBTAL returns immediately.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

STBTAL is called from DRLKON and PNTERR via:

```
8        16
TSX1     STBTAL
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

AU - Tally stored in second word.

SUPPORTING INFORMATION

Programming Method

STBTAL is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

TRANSMIT DATA TO BUFFER

STOBUF (EP6 of TSSK) stores the input data in the buffer.

PRECALLING SEQUENCE

Prior to entering STOBUF, the registers listed must contain the data indicated.

A	Input tally word (TALLYB)
Q	Character string to be added

CALLING SEQUENCE

STOBUF is called from the DRLKON and PNTERR via:

```
8       16
TSX1    STOBUF
Return  (input tally not run out)
Return  (both tallies run out)
Return  (data in buffer)
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

STOBUF is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

Locations, .TCTM3, 4, and 5 are used to store tallies and insert character string.

Other Routines Used

None.

FILE SPACING

TSXSPF (EP7 of TSSK) performs file spacing for routines within the TSS Executive. It sets the pointers to number of records, status return and file ID, then goes to DRLSPF to do actual spacing.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TSXSPF is called from PASFIL, TASK, EXENTR and DRLABT via:

```
 8        16
|        |
|TSX1    |TSXSPF
|ZERO    |L(file ID), L(number of 320-word blocks
|        |               to be spaced)
|ZERO    |L(status), 0
|Return  |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to DRLSPF.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

TSXSPF is nonreentrant.

Interrupts are not inhibited.

Uses common code in DRLSPF.

Storage

No internal temporary storage is used.

Other Routines Used

None.

SEARCH THE AFT

SRCH (EP8 of TSSK) searches the AFT to locate a specific file.

PRECALLING SEQUENCE

AQ contains file name in ASCII.

CALLING SEQUENCE

SRCH is called from EXENTR, SAVE, TASK, PASFIL, DRLPPT, TSXDIO, DRLDIO, DRLSPF, DRLFIL, DRLRFL, FILACT, MORLNK, PASFIL, DRLABT, RESTOR, PART, GROW and DRLSWH via:

```
8        16
TSX7     SRCH
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to calling routine.

POSTCALLING SEQUENCE

A   contains offset to PAT pointer,
    PAT pointer (contains permissions)

A ≠ 0 if file is found
  = 0 if file is not found

SUPPORTING INFORMATION

Programming Method

SRCH is nonreentrant and relocatable.

Interrupts are inhibited during master mode processing.

Storage

Internal temporary storage is used for storing registers.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

SWITCH FILE NAMES

TSXSWH (EP9 of TSSK) performs switching of names assigned to two temporary files. It stores the pointers and goes to DRLSWH for the actual switching of names.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TSXSWH is called by PATDEA when a user is logging-off.

```
 8       16
 TSX1    TSXSWH
 ZERO    L(filename 1),L(filename 2)
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

POSTCALLING SEQUENCE

A register contains error return, if any:

0 = File not open
1 = File not temporary

SUPPORTING INFORMATION

Programming Method

TSXSWH is nonreentrant and relocatable.

Interrupts are inhibited during master mode processing.

Other Routines Used

None.

DISC I/O DERAIL

DRLDIO (DRL DIO, TSSK) is used for files that appear in a user's Available File Table (AFT). It takes the equivalent of the calling sequence to GENIOS and performs the indicated seek, read, or write, using the master mode routines, .QUEUE and .LINK. It notifies the time-sharing system Allocator that I/O is being done, and to suspend subsystem execution, but not to swap. When the I/O is completed, the Allocator will again be notified, and the subsystem made eligible for execution. All hardware status is returned except device-busy, as such. Undefined file is returned as device-busy status (major status 01) and logical end of file as major status 17. If the terminal user does not have the necessary permissions, or if an invalid relative block number is given for a random file, the requesting subsystem is aborted and an error message is sent to the terminal.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLDIO is called from any TSS subsystem through EXDRL via:

```
8          16
DRL        DIO
SEEK command
ZERO       L(file ID),L(DCW1)
READ/WRITE command
ZERO       L(file ID),L(DCW2)
ZERO       L(status),0
Return
```

where:

fileid (2 words) contains the filename in ASCII - 1 to 8 characters.

dcw1 - IOTD L(rbn),1

where rbn contains, for random files, the relative block number (set by user). This word is always altered by I/O routines.

dcw2 - IOTD L(data),n

where data contains the starting address at which data is to be read/written, and n is the number of words to be transferred.

I/O commands - The user need not be concerned about giving commands for a specific device type. The seek (34) command, write (31) command, and read (25) command will be accepted for all devices. The actual commands used will be acquired from the channel module for the particular device.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to the calling subsystem if validation error or through LINSRV via Courtesy Call if I/O is issued.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

DRLDIO verifies that the user has the appropriate permission before he is allowed to read or write. In order to handle the EXECUTE permission, some additional tests must be performed.

If a file read is requested and the open file does not have READ permission, a check will be made to see if the requesting subsystem is one of the following processors:

BASIC (BASY)

FORTRAN (RUN)

CARDIN (CDIN)

If it is one of the above, a further check will be made to see if EXECUTE permission was granted on the opened file and if so the file read will be done. A read from one of the above subsystems is done only as the result of a RUN command.

Programming Method

DRLDIO is nonreentrant and relocatable.

Interrupts are inhibited while I/O is being done.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)
Line Service LINSRV (EP1 of TSSJ)
End Courtesy Call ENCC (EP5 of .MDISP)
Print Error Message PNTERR (EP1 of TSSG)
Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Assign an I/O Entry QUEUE (EP4 of .MIOS)
Link I/O to End of Queue LINK (EP1 of .MIOS)
Relinquish RLC (EP4 of .MDISP)
Search the AFT SRCH (EP8 of TSSK)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

DRLKOT(DRL KOUT) DRLKOT(DRL KOUT)
TSSK TSSK

KFYBOARD OUTPUT

DRLKOT (DRL KOUT, TSSK) accumulates the user's output in a buffer for eventual output to the keyboard. When the buffer is full, output to the keyboard is initiated. At this point, execution of the subsystem is inhibited and the subsystem made eligible for swap. When the output is complete, the program is again eligible for execution.

PRECALLING SEQUENCE

Prior to entering DRLKOT, the A-register will contain the KOUT/IN indicator if entry is from DRLKON.

CALLING SEQUENCE

DRLKOT is called from any TSS subsystem through EXDRL via:

```
 8         16
|          |
| DRL      |KOUT
| ZERO     |L(tally),L(char)
| Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The field L(tally) points to a driver tally word pointing in turn to a list of line TALLYB words which define each line of output ASCII characters to be sent to the keyboard. The driver tally has the count of the line tallies in the list. This allows the user to define scattered lines, not necessarily starting at word boundaries. The derail processor utilizes the tally words and they may be modified on return to the subsystem.

The optional field L(char) points to a word containing up to four characters that will be appended to the end of the output defined by each line tally. These characters could be line feed, carriage return, etc. If this field is not present in the calling sequence, no characters will be added. If the field is present, the first zero character terminates the appending of characters. In any case, no more than four characters will be appended. Note: Because of timing considerations and character set differences between terminal types, the carriage return characters should always precede the line feed character and one or two delete characters.

Programming Method

DRLKOT is nonreentrant.

Interrupts are inhibited during output.

Storage

No internal temporary storage is used.

Other Routines Used

Set Tally for Output Buffer STBTAL (EP5 of TSSK)
Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Line Service LINSRV (EP1 of TSSJ)
Transmit Data to Buffer STOBUF (EP6 of TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
Pass AFT to a Subsystem PASAFT (DRL PASAFT, TSSK)
Paper Tape Courtesy Call PPTCC1 (TSSJ)

KEYBOARD OUTPUT / INPUT

DRLKON (DRL KOUTN, TSSK) sends output to the keyboard device and holds the line open for a reply. It differs from DRL KOUTN only in that a reply is expected from the keyboard device.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLKON is called from any TSS subsystem through EXDRL via:

```
 8          16
 DRL        KOUTN
 ZERO       L(tally),L(char)
 Return
```

See DRLKOT for more details.

RETRIEVE LAST LINE OF INPUT

DRLKIN (DRL KIN, TSSK) retrieves the last line of input. It normally follows the KOUTN sequence. This is not necessary, however, and the DRL KIN may be repeated as many times as desired in repetitive processes such as those performed by EDIT, LIST, or BASIC. The last line will remain in the buffer until some output or additional input destroys it.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLKIN is called from any TSS subsystem through EXDRL via:

```
 8           16
|           |
|DRL        |KIN
|ZERO       |L(data),L(count of chars. to be transmitted
|           |          or zero if there is no data)
|ZERO       |L(status)  (see DRLKOT)
|Return     |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRLKIN is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

RETURN FROM SUBSYSTEM

DRLRET (DRL RETURN, TSSK) informs the Executive that this subsystem process has reached a normal termination. TSSH selects the next primitive in the sequence defined in the program descriptor and, based on this primitive, initiates the next process.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLRET is called from any TSS subsystem through EXDRL via:

```
 8         16
 DRL       RETURN
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRLRET is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EPl of TSSL)
Interpret Primitive SCAN3 (EP3 of TSSH)
Line Service LINSRV (EPl of TSSJ)
Buffer Dump BUFDMP (EP4 of TSSK)

DEFINE A TEMPORARY FILE

DRLFIL (DRL DEFIL, TSSK) obtains space for a requested temporary file, builds a PAT entry, and enters the file in the user's AFT.

PRECALLING SEQUENCE

Prior to entering DRLFIL, .TFTEM must contain the device type for temporary files.

CALLING SEQUENCE

DRLFIL is called from any TSS subsystem via:

```
 8        16
 DRL      DEFIL
 ZERO     L(File ID), L(Status)
 Return
```

where:

FILE ID

	0 – 23	24 – 35
	filename (chars. 1-4)	Status
+1	filename (chars. 5-8)	

	0 – 5	6 – 16	17	18	19 – 35
+2	Device Type		X	X	Number of links desired (binary)

Status (in binary) may be:

0 = Successful
3 = No room in AFT
4 = Temporary file not available
5 = Duplicate file name
6 = No room in PAT
7 = Illegal device specified

Device type is:

00 - DSU270
01 - DSU200
02 - DSU167
03 - MDU200
04 - MSS800
05 - DSS170
06 - DSS180

Bit 17 = 0	Use the system file device defined in .TFTEM.
= 1	Use device specified in ID+2.
Bit 18 = 0	Linked file
= 1	Random file

OPERATING SYSTEM INTERACTION

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

Upon successful return, A contains the file description, as follows:

Bits	
0-5	Device type (as above)
6-17	Number of words in block
18=0	
24-35 =	Number of links in file
or	
18=1	
24-35 =	Number of blocks in file
19=0	Linked file
19=1	Random file
20=0	Temporary file
20=1	Permanent file
21-23	Unused

This information is also returned when an already defined file is requested.

SUPPORTING INFORMATION

When a temporary file is requested, bit 17 of LOC+2 is examined to determine if the file is to be a specific device. If the requested file is to be on the system file, the device type is defined in .TFTEM. This word resides in the time-sharing Communication Region and can be changed according to the requirements of a particular site. If a specific device is requested but there is not enough space on that device, the request will be satisfied automatically from the system device.

Programming Method

DRLFIL is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Process Specific Device Request ENTRY (EP1 of .MALC2)
File Entry Space Allocation PATMAN (EP1 of TSSD)
Return LINKS/LLINKS to the Pool ONE (EP1 of .MALC9)

ABORT USER

DRLABT (DRL ABORT, TSSK) aborts the user when an abnormal sequence occurs during processing.

If the user has defined a file with the name ABRT, a core dump of the subsystem will be written to that file. In any case, a message will be sent to the terminal stating that an abort has occurred and the user is free to select a new system.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLABT is called from any TSS subsystem through EXDRL via:

8	16
DRL Return	DRLABT

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to calling subsystem.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRLABT is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
File Spacing TSXSPF (EP7 of TSSK)
Line Service LINSRV (EP1 of TSSJ)
Print Error Message PNTERR (EP1 of TSSG)
End Courtesy Call ENCC (EP5 of .MDISP)
Special Entry PNTTSX (EP2 of TSSG)

SET SWITCH WORD

DRLSET (DRL SETSWH, TSSK) sets the bits in the 36-bit switch word (.LSWTH) provided in each of the UST blocks associated with an individual user.

PRECALLING SEQUENCE

Prior to entering DRLSET, the registers listed must contain the data indicated.

Q Contains respective bits which are to be set in the switch word .LSWTH (see UST).

CALLING SEQUENCE

DRLSET is called from any TSS module through EXDRL via:

8	16
DRL Return	SETSWH

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

Q Contains switch word

SUPPORTING INFORMATION

The value of the Q-register is OR'd into the switch word. Thus, any bit that is "on" or "true" will be set true in the switch word. Other bits will not be disturbed. The value returned in the Q-register is the resultant setting of the switch word. This provides a method of reading the switch word and determining a course of logic based on events in preceding subsystem processes. Thus, if one subsystem process encounters an abnormal situation, a prearranged bit may be set and subsequent subsystems may interrogate the switch word and take appropriate action. In order to obtain some measure of discipline in the use of this feature, the first 18 bits (0-17) are reserved for usage by systems issued with the time-sharing system. The lower 18 bits are free for use by subsystems generated by users.

Programming Method

DRLSET is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

RESET SWITCH WORD

DRLRST (DRL RSTSWH, TSSK) turns off each bit in the switch word.

PRECALLING SEQUENCE

Prior to entering DRLRST, the registers listed must contain the data indicated.

Q contains respective bits which are to be rest in the switch word .LSWTH (See UST).

CALLING SEQUENCE

DRLRST is called from any TSS subsystem through EXDRL via:

8	16
DRL Return	RSTSWH

See DRLSET for further details.

REWIND FILE

DRLREW (DRL REW, TSSK) rewinds the specified linked file. It checks the calling sequence, sets the rewind flag and then transfers to the Forward/Backspace File routine for the rest of the processing. A request to rewind a random file causes a subsystem abort and an error message is sent to the terminal.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLREW is called from any TSS subsystem through EXDRL via:

```
 8       16
 DRL     REW
 ZERO    L(File ID),L(Status)
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to DRLSPF.

POSTCALLING SEQUENCE

Rewind flag is stored in SPFLAG.

SUPPORTING INFORMATION

Programming Method

DRLREW is nonreentrant and relocatable.

Interrupts are not inhibited.

Uses common code in DRLSPF.

Storage

No internal temporary storage is used.

Other Routines Used

None.

FORWARD / BACKSPACE A FILE

DRLSPF (DRL FILSP, TSSK) spaces a linked file forward or backward n blocks. If a request is made to space a random file, the requesting subsystem is aborted and an error message is sent to the terminal.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLSPF is called from any TSS subsystem through EXDRL via:

```
  8       16
 ---------------------------------------------------------------
|         |
| DRL     |FILSP
| ZERO    |L (File ID),L(number of 320-word blocks to be
|         |             spaced.  Negative number indicates
|         |             backspace.)
| ZERO    |L (Status), 0
| Return  |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

```
          TRA      2,1   (if call was TSX to TSXSPF) or
          TSX1     RETSBS
```

POSTCALLING SEQUENCE

Possible status returns:

Loadpoint (0002)
Logical End-of-file (1700)
Device Busy, Undefined File (01)

SUPPORTING INFORMATION

Programming Method

DRLSPF is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Print Error Message PNTERR (EP1 of TSSG)
Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Search the AFT SRCH (EP8 of TSSK)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

RELEASE A FILE

DRLRFL (DRL RETFIL, TSSK) releases (temp) or deaccesses (perm) a specified file or all files except SY**.

When a temporary file is returned, the file space and PAT entry space is released and the file deleted from the AFT. When a permanent file is returned, the file system is notified to deaccess the file, the PAT entry space is released, and the file deleted from the AFT.

Note: A file that cannot be found in the AFT is considered by DRLRFL to be already released.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLRFL is called from any TSS subsystem through EXDRL via:

```
 8       16
 DRL     RETFIL
 ZERO    L(File ID), L(Buffer)
 Return
```

where:

File ID (2 words) contains the file name in ASCII, or a right-justified 777 (in word 1) if all files (except SY**) are to be returned.

buffer (BSS 380) is a work area used by the system; this area is not required for temporary files.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRLRFL is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Search the AFT SRCH (EP8 of TSSK)
File Deallocation PATDEA (EP2 of TSSD)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

RELMEM(DRL RELMEM)
TSSK

RELMEM(DRL RELMEM)
TSSK

RELEASE MEMORY

RELMEM (DRL RELMEM, TSSK) reduces the size of core required by a subsystem execution. It can also be used to change the register storage location only, without releasing memory.

PRECALLING SEQUENCE

Prior to entering RELMEM, the registers listed must contain the data indicated.

A - Return location, register-storage location
Q - Number of words low, number of words high

CALLING SEQUENCE

RELMEM is called from any TSS subsystem through EXDRL via:

```
8        16
DRL      RELMEM
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The number of words to be released from the lower portion of the subsystem is in the left-half of the Q-register and the number of words to be released from the upper portion of the subsystem is in the right-half of the Q-register. Both these values must be in blocks of 1024 words or will be used mod 1024.

Programming Method

RELMEM is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for storing registers.

Other Routines used

Update Allocator Queue ALLOC (EP1 of TSSL)

ADD MEMORY

ADDMEM (DRL ADDMEM, TSSK) requests additional core required by a subsystem program during execution.

PRECALLING SEQUENCE

Prior to entering ADDMEM, the registers listed must contain the data indicated.

A - Return location, register storage location
Q - 0, number of words high

CALLING SEQUENCE

ADDMEM is called from any TSS subsystem through EXDRL via:

```
 8         16
┌──────────┬──────────────────
│ DRL      │ADDMEM
│ Return   │
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The value in Q is interpreted as a request for additional memory. Only high memory may be requested. If memory is not available adjacent to the requesting subsystem, the subsystem will be swapped out. It is not possible to add memory to the lower end of a subsystem region.

Programming Method

ADDMEM is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)

CORFIL(DRL CORFIL) TSSK

CORFIL(DRL CORFIL) TSSK

READ / WRITE CORE FILE

CORFIL (DRL CORFIL, TSSK) passes data from one subsystem to another without accessing a mass storage device. The core file is ten words of the user status table space.

PRECALLING SEQUENCE

Prior to entering CORFIL, the registers listed must contain the data indicated.

```
A -  0-17 - Location within subsystem where data is to be read in
            or written out
    18-35 - Number of words to be transmitted

Q -  0-17 - Number of core file cell at which transmission is to
            begin
    18-35 - 0 = Subsystem to core file
            1 = Core file to subsystem
```

CALLING SEQUENCE

CORFIL is called from any TSS subsystem through EXDRL via:

```
 8        16
|DRL     |CORFIL
|Return  |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

CORFIL is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Print Error Message PNTERR (EP1 of TSSG)

OBTAIN SNUMB

SNUMB (DRL SNUMB, TSSK) assigns SNUMB's to batch jobs spawned by subsystem programs. The numbers range from 1 to 7777 (octal). When 7777 is reached, the sequence begins again. The initial SNUMB is set by TSTART according to time of day at startup.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

SNUMB is called from any TSS subsystem through EXDRL via:

```
 8        16
 DRL      SNUMB
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

A contains the SNUMB in BCI format.

SUPPORTING INFORMATION

Programming Method

SNUMB is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

OBTAIN PROCESSOR TIME, DATE, TIME OF DAY

TIME (DRL TIME, TSSK) returns the processor time used and time of day. It also returns the date upon request.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TIME is called from any TSS subsystem through EXDRL via:

```
  8       16
 DRL      TIME
 ZERO     L(date) (If zero, date is not returned)
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

A - Sum of processor time for user (1/64 ms)
Q - Time of day (1/64 ms from midnight)
L date - MM/DD/YY in ASCII, if requested

SUPPORTING INFORMATION

Programming Method

TIME is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Provide Date and Time of Day FGTIM (EP8 of .MFALT)

PASS AFT TO A SUBSYSTEM

PASAFT (DRL PASAFT, TSSK) places the names of the files in the user's AFT in the area specified and returns it to the subsystem.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

PASAFT is called from any TSS subsystem through EXDRL via:

```
 8         16
|DRL      |PASAFT
|ZERO     |L(aftarea),L(max. number of files to be
|         |             passed.  If zero, all files
|         |             are passed)
|Return   |
```

where:

aftarea is 2 X number of files to be passed, plus one, formatted as follows:

Word 1	- Number of files in AFT
Word 2	- Chars. 1-4 of first filename
Word 3	- Chars. 5-8 of first filename
.	
.	
.	
.	
Word n*2	- Chars. 1-4 of nth filename
Word n*2+1	- Chars. 5-8 of nth filename

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

PASAFT is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

TERMINAL TYPE

TERMTP (DRL TERMTP, TSSK) returns the terminal ID and type code to the calling routine.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TERMTP is called from any TSS subsystem through EXDRL via;

```
 8         16
|DRL      |TERMTP
|Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

A register contains (in BCl):

```
 0-17 - Zero
18-23 - Terminal type code
        (1-3 undefined, 4 TTY)
24-35 - Terminal ID
```

SUPPORTING INFORMATION

Programming Method

TERMTP is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

PRIVILEGED DISC I/O

PDIO (DRL PDIO, TSSK) performs I/O on time-sharing system files.

Files allocated directly to time-sharing Executive appear only in the .TCFIL table. In general, these system files are accessed only from within the Executive portion of the system. However, some of the executive functions of the system are performed by privileged subsystem programs and it is necessary for these subsystems to have access to system files. This is done through PDIO.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

PDIO is called from any privileged TSS subsystem through EXDRL via:

```
8       16
DRL     PDIO
SEEK command
ZERO    L(file ID), L(DCW1)
READ/WRITE command
ZERO    L(file ID), L(DCW2)
ZERO    L(status),0
```

where file ID is a 2-character BCI filename, right-justified with leading zeros. Other parameters are defined in DRLDIO.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Control transfers to DIOCOM in DRLDIO.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

PDIO is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

OVERLAY - LOAD A SUBSYSTEM OR PERM FILE

RESTOR (DRL RESTOR, TSSK) loads a subsystem or part or all of the contents of a permanent file as an extension or an overlay of the calling subsystem. When the initial load address is nonzero, it is loaded at LOC. This allows loading of floatable libraries, etc., without having to provide space for the data area.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

RESTOR is called from any TSS subsystem through EXDRL via:

```
8          16
DRL        RESTOR (subsystem name)
ASCII      1, filename
ZERO       Location at which to load (LOC), zero or
              initial load location of subsystem
ZERO       Location of next instruction, zero
Return
```

```
DRL        RESTOR (PERM file)
ZERO       Name pointer, 0 or 1*
ZERO       Location at which to load (LOC), zero or
              initial load location of PERM file
ZERO       Location of next instruction, location of buffer
Return
```

*0 indicates file name = program name (i.e., only one program on the perm file); buffer = 40 words on a DSU204, 64 words on other mass storage devices.

1 indicates the name pointer designates three words -- a 1-8 character ASCII name in the first two words and a 1-6 character BCD program name in the third word; buffer = 64 words.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RESTOR is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Print Error Message PNTERR (EP1 of TSSG)
MME GEINOS Processor INOS (EP5 of .MIOS)
System Error ERROR (EP1 of TSSF)
Update Allocator Queue ALLOC (EP1 of TSSL)
End Courtesy Call ENCC (EP5 of .MDISP)
Special Entry PNTTSX (EP2 of TSSG)
Search the AFT SRCH (EP8 of TSSK)
Executive Disc I/O TSXDIO (EP3 of TSSK)

PASS FILE TO BATCH PROCESSOR

PASFIL (DRL SPAWN, TSSK) generates the information required by GEIN to process this job.

PRECALLING SEQUENCE

Prior to entering PASFIL, the registers listed must contain the data indicated.

```
     A = 0, Immediate return to subsystem
         1, Return after batch job complete
    QU   Filename pointer
```

CALLING SEQUENCE

```
         8         16
        ┌─────────┬──────────────────────────────────
        │ DRL     │SPAWN
        │ ZERO    │SNUMB pointer, buffer pointer
        │ Return  │
        │         │
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRLAL gate is closed while searching the .CRSNB table for duplicate SNUMB and program number. It is opened before relinquishing. The .CRPOQ gate is shut while testing and resetting tally.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

QU contains error codes as follows:

0 = No error
1 = Undefined file
2 = No PAT for PASFIL
3 = Duplicate SNUMB, resubmit with new SNUMB
4 = SNUMB not given
5 = No program number available, try again

SUPPORTING INFORMATION

The calling subsystem has written the file as a stacker input job deck on a linked file, the name of which is pointed to by QU. The file must have an empty 320-word first block. The input deck (beginning with the second block numbered "1") must have SNUMB and ENDJOB records as in a normal input deck. The file is in BCI format.

The SNUMB pointer specifies the location of the SNUMB obtained via DRL SNUMB. The buffer pointer points to a 325-word work area provided by the subsystem. PASFIL generates the information required by GEIN in this area and writes it into the 320-word first block of the files.

Upon return to the subsystem, any error condition is indicated by one-character BCI error code, right-justified in QU.

Programming Method

PASFIL is nonreentrant and relocatable.

Interrupts are inhibited during master mode processing.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP3 of TSSD)
Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Executive Disc I/O TSXDIO (EP3 of TSSK)
Search the AFT SRCH (EP8 of TSSK)
File Spacing TSXSPF (EP7 of TSSK)
Line Service LINSRV (EP1 of TSSJ)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Relinquish RLC (EP4 of .MDISP)
Enable Program ENB (EP6 of .MDISP)
End Courtesy Call ENCC (EP5 of .MDISP)
Obtain Normal Snapshot GESNIP (EP1 of .MSNP1)
Delinking a File FREE (EP3 of TSSD)

DRLPPT(DRL TAPEIN) DRLPPT(DRL TAPEIN)
TSSK TSSK

PAPER TAPE INPUT

DRLPPT (DRL TAPEIN, TSSK) starts paper tape input from a teletypewriter. The TSS Executive builds the input onto the TAP* file.

PRECALLING SEQUENCE

Prior to entering DRLPPT, the registers listed must contain the data indicated.

A File description of TAP* file

CALLING SEQUENCE

DRLPPT is called from any TSS subsystem through EXDRL via:

```
8         16
DRL       TAPEIN
ZERO      L(tally),L(Char)   (same as DRLKOT)
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRLPPT is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Print Error Message PNTERR (EP1 of TSSG)
Search the AFT SRCH (EP8 of TSSK)

INTERNAL CALL TO ANOTHER SUBSYSTEM

CALLSS (DRL CALLSS, TSSK) swaps the calling subsystem to the swap file, to be resumed later when a POPUP primitive of the called subsystem is executed. A pushdown list of internally-called subsystems is a part of each UST. Information required to reload and resume processing of the calling subsystem is stored in 2-word entries. Internal calls may be no more than 3 deep. (The internal-call pushdown list is distinct from the program stack, also in the UST.)

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

CALLSS is called from any TSS subsystem through EXDRL via:

```
 8        16
 DRL      CALLSS
 ASCII    1,name of subsystem to be called
   .
   .
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

CALLSS is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)
Print Error Message PNTERR (EP1 of TSSG)
TSS Subsystem Startup STARTP (EP5 of TSSI)

USER IDENTIFICATION

USERID (DRL USERID, TSSK) stores the information supplied by the terminal user during log-on into the UST and increments the return address by one. If the information is incorrect, an error message is sent to the terminal. The eighth word of the user's SMC is also picked through a pointer and stored in .LACPT of the UST.

PRECALLING SEQUENCE

Prior to entering USERID, the registers listed must contain the data indicated.

```
AQ     User ID in BCI
X0     0-5  Priority
       6-17 Disc address
```

CALLING SEQUENCE

USERID is called from the NEW subsystem during the HELLO sequence via:

```
 8        16
-----------------------------------------------
|DRL      |USERID
|ZERO     |Pointer to word 7 of SMC, 0
|  .      |
|  .      |
|Return   |
```

Where:

Word 7 of the system master catalog is used to specify legal CARDIN and LODX users and a batch job urgency for CARDIN.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

Word 7 of SMC is stored in .LACPT of UST.

SUPPORTING INFORMATION

Programming Method

USERID is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Print Error Message PNTERR (EP1 of TSSG)

USER TERMINATE

TERMPG (DRL TERMPG, TSSK) resets the UST to its initial condition. It is normally called only from the TERM subsystem. If another subsystem does call, the derail is accepted but the terminate function is bypassed and no message is issued. LINSRV performs no more I/O to the designated line.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TERMPG is called from the TERM subsystem via:

```
  8          16
 ┌──────────┬─────────────────
 │ DRL      │TERMPG
 │   .      │
 │   .      │
 │ Return   │
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

TERMPG is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Release Buffer TERM76 (TSSO)
Release UST RUST (EP3 of TSSN)
Line Service LINSRV (EP1 of TSSJ)
Terminate Routine TERM (EP1 of TSSI)
Clear UST and Make Accounting Entry TRMCU (EP2 of TSSI)

PASS UST TO SUBSYSTEM

PASUST (DRL PASUST, TSSK) copies the designated UST to the buffer provided by the subsystem.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

PASUST is called from any TSS subsystem through EXDRL via:

```
 8         16
 ___________________________________________
|          |
| DRL      |PASUST
| ZERO     |L(buffer), number of words to pass
|    .     |
|    .     |
| Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

PASUST is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

ADD LINKS TO TEMPORARY FILE

MORLNK (DRL MORLNK, TSSK) acquires the additional number of links requested, if possible, and updates the PAT entry for the file.

PRECALLING SEQUENCE

Bits 0-5 of LINKS will be masked before the request for more space is processed.

CALLING SEQUENCE

MORLNK is called from any TSS module through EXDRL via:

```
 8         16
|DRL       |MORLNK
|ZERO      |L(LINKS),L(fileID)
|Return    |
```

where

```
     LINKS   0-17 - Number of additional links requested
            18-35 - On return, number of links obtained

     file ID      - 2-word filename in ASCII
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

X4 - contains number of links obtained
(number requested or zero)

LINKS 0-5 contain a 1 in the corresponding bit position if one of the following errors occur:

0 - PAT full
1 - Space exhausted
2 - PERM file
3 - File not in AFT
4 - No links requested
5 - Not used

SUPPORTING INFORMATION

Programming Method

MORLNK is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for buffer.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
File Entry Space Allocation PATMAN (EP1 of TSSD)
Delinking a File FREE (EP3 of TSSD)
Additional Mass Storage Request MINK (EP2 of .MMORE)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

NEWUSR(DRL NEWUSR) NEWUSR(DRL NEWUSR)
TSSK TSSK

LOG-ON NEW USER WITHOUT DISCONNECT

NEWUSR (DRL NEWUSR, TSSK) allows one user to log off and another to log on without disconnecting the terminal. The accounting functions are performed for the first user, the UST reinitialized and time reset for the new user. If the calling subsystem is not NEWU, the call is ignored.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

NEWUSR is called from the NEWU subsystem via:

```
8          16
DRL        NEWUSR
  .
  .
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

NEWUSR is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Terminate Routine TERM (EP1 of TSSI)
Reset Flags and Indicators NEWU76 (TSSO)
Line Service LINSRV (EP1 of TSSJ)
System Message Generator SYS (EP4 of TSSI)
Clear UST and Make Accounting Entry TRMCU (EP2 of TSSI)

PERMANENT FILE REQUESTS

FILACT (DRL FILACT, TSSK) processes all permanent file requests with the exception of file deaccess, which is handled by DRLRFL. The particular function requested is specified in the upper half of word 3 of the calling sequence. It also allows the master subsystem to purge another user's files for the delete SMC usage.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

FILACT is called from any TSS subsystem through EXDRL via:

```
 8        16
|        |
|DRL     |FILACT
|ZERO    |Alternate name, L(arglist)
|ZERO    |Function number, L(buffer)
|Return  |
```

```
 8        16
|        |
|DRL     |FILACT
|ZERO    |Pointer, L(arglist)
|ZERO    |Function number, L(buffer)
|Return  |
```

where:

Entries not required for a function must be 0.

Alternate name - Two-word entry used when a file is to be accessed by a file name other than that by which it was created (e.g., a file created in the batch environment with a name of more than 8 characters, or a file whose name is the same as one already in the user's AFT). Note: When an alternate name is used, the defined file name in the catalog/file description must be in BCI and the alternate name in ASCII. (Used by functions 4 and 5.)

Pointer - Pointer to the user-id of the user being deleted by the master subsystem.

Arglist -

```
ZERO  L(status),L(random flag)
ZERO  L(Cat/File Description),, L(permissions)
Zero  L(options), L(new name)
```

Used by functions as follows:

Status (two words) - All
Random flag - 4 and 5
Cat/File Description (n words) - All
Permissions (1 word) - 2, 3, 4, 10 and 11
Options (n words) - 2, 3, 10 and 11
New name (4 words) - 10 and 11

Function numbers

1 -	System master catalog entry create
2 -	Create catalog
3 -	Create file
4 -	Access file and set busy
5 -	Access file and do not set busy
8 -	Purge catalog
9 -	Purge file
10 -	Modify catalog
11 -	Modify file
12 -	System master catalog entry delete
13 -	System master catalog entry update
14 -	System master catalog entry query
15 -	System master catalog entry query and set busy
22 -	Release File

Function numbers 16-21 require the following arglist:

```
ZERO  L(status),L(record)*
Zero  L(Cat/File Description), device name*
      *Filled by file system
```

16 -	Get first master
17 -	Get next master
18 -	Get current
19 -	Get first
20 -	Get next
21 -	Get specific

Buffer - 380 words

This buffer is required by all functions. The first 25 words are used by the derail processor to build and execute the calling sequence to the GECOS File System. The remaining 355 words are used by the GECOS File System.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

An ACCESS on a file that is already open makes a permission check. When an attempt is made to write or read a file without proper permission, a status of 4037 (octal) is returned to the user which says that the file is already open, without regard to the new permissions being asked for.

On any access of an already open file, the new permissions being asked for must be equal to, or a subset of, those already granted on the open file. If the permissions are not equal to or a subset, a permissions denied or status 4003 (octal) is returned.

For example, if a file is open with READ only permission, a subsequent request for ACCESS with EXECUTE only specified will result in a status 4037, file already open.

If a file is open with EXECUTE only permission, a subsequent request for ACCESS with READ permission specified will result in a status 4003, permission denied.

Programming Method

FILACT is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for the buffer.

Other Routines Used

Disc I/O Derail DRLDIO (DRL DIO, TSSK)
File Entry Space Allocation PATMAN (EP1 of TSSD)
Update Allocator Queue ALLOC (EP1 of TSSL)
Line Service LINSRV (EP1 of TSSJ)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Print Error Message PNTER (EP1 of TSSG)
Print Error message PNTERR (EP1 of TSSG)
End Courtesy Call ENCC (EP5 of .MDISP)
Search the AFT SRCH (EP8 of TSSK)
Delinking a File FREE (EP3 of TSSD)

SET LINE NUMBER / INCREMENT IN UST

SETLNO (DRL SETLNO, TSSK) stores the specified line number and increment in the User Status Table for use by Line Service. An indicator (automatic line numbering mode) and a blank/no blank indicator are set in .TFLG2.

PRECALLING SEQUENCE

Prior to entering SETLNO, the registers listed must contain the data indicated.

```
A     Line number
Q     Increment
      Bit 0 in Q is the blank/no blank indicator
                (0=0=Blank,
                 0≠0=No blank)
```

CALLING SEQUENCE

SETLNO is called from any TSS subsystem through EXDRL via:

```
8         16
DRL       SETLNO
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

SETLNO is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

RETURN TO SYSTEM MODE

SYSRET (DRL SYSRET, TSSK) causes the subsystem to be killed and returns to the system selection point, bypassing the normal return via the primitives.

PRECALLING SEQUENCE

Prior to entering SYSRET, the registers listed must contain the data indicated.

A contains a one.

CALLING SEQUENCE

SYSRET is called from any TSS subsystem through EXDRL via:

```
 8         16
|DRL      |SYSRET
|Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to the system selection point.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

SYSRET is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)
System Message Generator SYS (EP4 of TSSI)
Line Service LINSRV (EP1 of TSSJ)
Buffer Dump BUFDMP (EP4 of TSSK)

STOP A MASTER SUBSYSTEM

STPSYS (DRL STPSYS, TSSK) removes a subsystem from the allocator's queue. It is used for those master subsystems which set indicators for the time-sharing Executive to receive output controlled by the executive functions only.

This derail should *not* be used by any subsystem which requires that control be returned and should only be used as a method of turning control of a particular remote over to the time-sharing Executive.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

STPSYS is called from any master TSS subsystem through EXDRL via:

```
8         16
DRL       STPSYS
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

STPSYS is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)

DISCONNECT TERMINAL

DRLDSC (DRL DRLDSC, TSSK) sends any output in the buffer to the terminal and, if necessary, sends the disconnect ICM to the DATANET 30.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLDSC is called from any TSS subsystem through EXDRL via:

```
 8         16
 ______________________________
|          |
| DRL      |DRLDSC
| Return   |
|          |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

DRLDSC gives the subsystem the ability to disconnect terminals which are not disconnected by receipt of the EOT character (004 ASCII). Any input which has been inserted into the line buffer area for the terminal is output and upon completion, a MME GEROUT disconnect is issued. The subsystem will not be reentered after execution of the DRLDSC.

Programming Method

DRLDSC is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Line Service LINSRV (EP1 of TSSJ)
Update Allocator Queue ALLOC (EP1 of TSSL)
Buffer Dump BUFDMP (EP4 of TSSK)
MME GEROUT Processor GROUT (EP1 of .MROUT)

PASS AFT NAMES AND DESCRIPTIONS

PASDES (DRL PASDES, TSSK) places the user's available file names and the one-word file descriptions in the area specified by the user.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

PASDES is called from any TSS subsystem through EXDRL via:

```
 8        16
|DRL      |PASDES
|ZERO     |L (buffer), 0
|Return   |
```

where

The maximum size of the buffer is 3 times the number of files in the AFT, plus 1 (.LFILES*3+1)

```
.LFILES     Contains the number of files in AFT

          ( Chars 1-4 of first file name
file 1   {  Chars 5-8 of first file name
          ( Description of file 1
                      .
                      .
                      .
          ( Chars 1-4 of nth file name
file n   {  Chars 5-8 of nth file name
          ( Description of nth file
```

The file description word is formatted:

0-5	Device type
6-17	No. of words per block
If 18=0	
24-35	No. of links per file
If 18=1	
24-35	No. of blocks per file
19=0	Linked file
19=1	Random file
20=0	Temporary file
20=1	Permanent file
21-23	Unused

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

Q-register contains the file description word.

SUPPORTING INFORMATION

Programming Method

PASDES is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

GET JOB STATUS

JSTS (DRL JSTS, TSSK) obtains the processing status of the batch job identified by SNUMB.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

JSTS is called from any TSS subsystem through EXDRL via:

```
8         16
DRL       JSTS
BCI       1, SNUMB (derail arg. word)
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

Derail arg word contains:

bits 0-8	Status
	0 = Output ready
	1 = Reading - card reader
	2 = Reading - magnetic tape
	3 = Reading - remote terminal
	4 = Waiting - allocation
	5 = Waiting - peripherals
	6 = Waiting - core
	7 = In hold
	8 = In limbo
	9 = Executing
	10 = Swapped out
	11 = Waiting - tape
	12 = Too big
	13 = Overdue
	14 = In restart
	15 = Terminating
	16 = Output waiting
	17 = Output complete
	18 = Not in system
9-17	Activity number
18-35	SNUMB

SUPPORTING INFORMATION

Programming Method

JSTS is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

PROCESS LINE SWITCH

CGROUT (DRL CGROUT, TSSK) switches a remote terminal line to direct-access connection with a slave program (Op=25), or switches any line(s) connected to a slave program back to TSS (Op=26), provided such lines were originally connected to TSS.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

CGROUT is called from any TSS subsystem through EXDRL via:

```
8        16
DRL      CGROUT
VFD      18/0, 6/Op, H12/line-ID
ZERO     L(SNUMB),0
Return
```

If Op=25, line-ID is the station code of line
          to be switched.

If Op=26, line-ID or zero, which implies all lines.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

On return (Op 26), line-ID equals zero if all lines connected to SNUMB have been reswitched; if not equal to zero, line-ID is the station code of a line still receiving output.

SUPPORTING INFORMATION

Programming Method

CGROUT is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

MME GEROUT Processor GROUT (EP1 of .MROUT)

PARTIAL RELEASE OF TEMPORARY FILES

PART (DRL PART, TSSK) releases a portion of a temporary file. If the specified number of links is greater than or equal to the number of links in the file, the file is reduced in size to one link. When the request has been satisfied, the file is in the rewound position.

POSTCALLING SEQUENCE

None.

CALLING SEQUENCE

PART is called from any TSS subsystem through EXDRL via:

```
 8        16
 DRL      PART
 ZERO     L(file name), number of links to be released
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Returns to RETSBS.

POSTCALLING SEQUENCE

A-register contains status as follows:

AU=0	Request satisfied
AU=1	File nonexistent or not a temporary file.

SUPPORTING INFORMATION

Programming Method

PART is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
Partial Release of Mass Storage File CALL (EP2 of .MRELS)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

INCREASE PERM FILE TO MAXIMUM SIZE

GROW (DRL GROW, TSSK) increases the size of an already accessed (open) permanent file up to its specified maximum file.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

GROW is called from any TSS subsystem through EXDRL via:

```
 8        16
 -------------------------------
|        |
| DRL    |GROW
| ZERO   |L(n),L(file name)
| ZERO   |L(buffer),L(status)
| Return |
|        |
```

where n may be

0 to allow the system to determine size of the increment, or the number of 320-word blocks to be added to the file.

Buffer is a 158-word work area.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

Bits 0-11 of the status word may contain:

4000	No errors
4002	I/O error - cannot proceed
4010	Link space exhausted
4020	Filename illegal
4024	Internal link table checksum error

SUPPORTING INFORMATION

Programming Method

GROW is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
Extend Size of TSS File ENT2 (EP2 of .MFSI9)
End Courtesy Call ENCC (EP5 of .MDISP)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

ABORT A JOB

ABTJOB (DRL ABTJOB, TSSK) aborts the batch job identified by SNUMB and submitted from the requesting terminal.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

ABTJOB is called from any TSS subsystem through EXDRL via:

```
8          16
DRL        ABTJOB
BCI        1,SNUMB
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

The derail argument word contains, in place of SNUMB:

0 = Job not in system
1 = Job not initiated from requesting terminal
    (not your job)
2 = Abort initiated

SUPPORTING INFORMATION

Programming Method

ABTJOB is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Process KILL Typein KILL (EP1 of .MPOP1)

MASTER TTY/600 CONSOLE TALK

CONSOL (DRL CONSOL, TSSK) permits a master user to talk to the GE-600 operator's console and receive a reply.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

CONSOL is called from any TSS master subsystem through EXDRL via:

```
 8          16
|DRL       |CONSOL
|ZERO      |Output buffer, response indicator
|ZERO      |Input buffer
|Return    |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV to continue I/O; to RETSBS to return to caller.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

CONSOL is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Disc I/O Derail DRLDIO (TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
MME GEINOS Processor INOS (EP5 of .MIOS)
End Courtesy Call ENCC (EP5 of .MDISP)

SWITCH NAMES ASSIGNED TO TEMPORARY FILE SPACE

DRLSWH (DRL SWITCH, TSSK) switches the names assigned to two temporary files. This effectively exchanges the contents of the two files.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DRLSWH is called from any TSS subsystem through EXDRL via:

```
8        16
DRL      SWITCH
ZERO     L(file name 1),L(file name 2)
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

A register contains error return, if any:

0 = File not open
1 = File not temporary

SUPPORTING INFORMATION

Programming Method

DRLSWH is nonreentrant and relocatable.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

TIME AND SIZE LIMIT

DRLIMT (DRL DRLIMT, TSSK) stores the time and size limits in the UST.

PRECALLING SEQUENCE

Prior to entering DRLIMT, the registers listed must contain the data indicated.

A - Time limit

CALLING SEQUENCE

DRLIMT is called from any TSS subsystem through EXDRL via:

```
8         16
|DRL      |DRLIMT
|Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRLIMT is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

Other Routines Used

None.

BATCH OUTPUT REQUEST

JOUT (DRL JOUT) is a privileged derail which can be executed only by the JOUT subsystem. It provides an interface with .MGEOT and allocates, deallocates and updates PAT bodies for the requesting JOUT subsystem.

PRECALLING SEQUENCE

Prior to entering JOUT, the registers listed must contain the data indicated.

X2	UST address
X4	TSS relative arglist
X3	DRL location

CALLING SEQUENCE

JOUT is called from the JOUT subsystem through EXDRL via:

```
8       16
DRL     JOUT
ZERO    OPCODE,ARGLIST
ZERO    STATUS,0
Return
```

The arglist is dependent on the op code as follows:

```
Op Code                 Arg List

  01 - Get SNUML Entry
                        BCI  1,SNUMB
                        ZERO 0,0
                        ZERO 0,16

  02 - Assign a PAT Body
                        ZERO 0,0

  03 - Update PAT (Regular)
                        ZERO  Offset to PAT Body, File Code
                        ZERO  SYSOUT Link#, Device
```

```
    04 - Print at Central Site
    05 - Direct Output to Remote
    06 - Flush Output
    07 - Deaccess Output
         (Directives to SYSOUT to manipulate output)

    08 - Deallocate PAT

    09 - Update PAT (Special)
                        ZERO  Offset to PAT pointer, File Code
                        ZERO  SCT, Absolute Link#
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRSYQ gate is used for communication with GEOT.

ROUTINE RETURNS

```
       8        16
      ┌─────────┬──────────────────────
      │ TSX1    │RETSBS
      │         │
```

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

JOUT is reentrant and written in floatable code.

Interrupts are inhibited while assigning a PAT body and storing registers.

Storage

Internal temporary storage is used for saving registers during the call to .MDISP.

Other Routines Used

Update Allocator Queue ALLOC (EP1 of TSSL)
Line Service LINSRV (EP1 of TSSJ)
Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Print Error Message PNTERR (EP1 of TSSG)
Enable Program ENB (EP6 of .MDISP)

KEYBOARD OUTPUT FROM UNFILLED BUFFER

OUTNOW (DRL KOTNOW, TSSK) forces output of a partially full buffer. It differs from DRL KOUT in that short, intermittent messages from a subsystem are not stacked up waiting for the buffer to fill. KOTNOW retains control until the buffer is flushed.

PRECALLING SEQUENCE

Prior to entering OUTNOW, the A-register will contain the indicator if entry is from DRLKON.

CALLING SEQUENCE

OUTNOW is called from any TSS subsystem through EXDRL via:

```
8         16
DRL       KOTNOW
ZERO      L(tally),L(char)
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The field L(tally) points to a driver tally word pointing in turn to a list of line TALLYB words which define each line of output ASCII characters to be sent to the keyboard. The driver tally has the count of the line tallies in the list. This allows the user to define scattered lines, not necessarily starting at word boundaries. The derail processor utilizes the tally words and they may be modified on return to the subsystem.

The optional field L(char) points to a word containing up to four characters that will be appended to the end of the output defined by each line tally. These characters could be line feed, carriage return, etc. If this field is not present in the calling sequence, no characters will be added. If the field is present, the first zero character terminates the appending of characters. In any case, no more than four characters will be appended. Note: Because of timing considerations and character set differences between terminal types, the carriage return characters should always precede the line feed character and one or two delete characters.

Programming Method

OUTNOW is nonreentrant.

Interrupts are inhibited during output.

Storage

No internal temporary storage is used.

Other Routines Used

Set Tally for Output Buffer STBTAL (EP5 of TSSK)
Return to Subsystem from Derail RETSBS (EP2 of TSSK)
Line Service LINSRV (EP1 of TSSJ)
Transmit Data to Buffer STOBUF (EP6 of TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
Pass AFT to Subsystem PASAFT (DRL PASAFT, TSSK)
Paper Tape Courtesy Call PPTCC1 (TSSJ)

PROCESSOR TIME AND CORE SIZE LIMIT

OBJTIM (DRL OBJTIM) sets bit 6 of the .LSWTH of the user's UST so that core size limit and processor time limit are checked.

CALLING SEQUENCE

OBJTIM is called from any TSS subsystem through EXDRL via:

```
8         16
|DRL      |OBJTIM
|Return   |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to RETSBS.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

1. Setting Limits

   a. Installation core size and processor time limits are assembled or patched into words .TASSZ and .TSTM of TSSA. The core size is put in .TASSZ in number of words. The time limit is put in .TASTM in pulses (number of seconds *64*1000).

   b. User-specified processor time limit is set by the BASIC and FORTRAN subsystems according to the user directive on the RUN command line. The user types:

      RUN-nn

      where nn is the object run time limit the user wishes in seconds. BASIC and FORTRAN place this time limit (in seconds) in the A-register and do a DRL DRLIMT. The derail processor converts this to pulses and places it in .LIMIT of the user's User Status Table.

2. Notifying the Time Sharing Executive that it is Generated Code Now Being Executed:

   The BASIC and FORTRAN compilers will generate, as the first instruction of the user program, a DRL OBJTIM. This derail sets bit 6 of .LSWTH of the user's UST. Unless this bit is set, neither core size limit nor processor time limit will be checked.

3. Checking Limits

   a. Processor Time Limits

      TSSM, if bit 14 of .LSWTH of the UST is on, sets a timer (.LIMTR in the UST), with the smaller of .TASTM (installation time limit) or .LIMIT (user time limit). Each time through TSSM, the timer (.LIMTR) is then decremented until it runs out or until the interaction ends.

   b. Core Size Limit

      The core size limit is checked in TSSL at two different times if bit 6 of .LSWTH is on.

      1) At the time of the DRL OBJTIM to see that the size is not already too big and

      2) At the time of an ADDMEM request by the subsystem.

4. Return

   When either size or time limits are exceeded, word 11 (decimal) of the subsystem prefix is checked for a fault vector. If word 11 contains a valid fault vector, the return to the subsystem is through this word. The IC and I where the fault occurred is stored in word 10 with bit 34 set for size limit exceeded and bit 35 set for time limit exceeded.

   Time-sharing will type the appropriate one of two messages if there is no fault vector and return to the user at system level. These messages are:

   064 - EXECUTE TIME LIMIT EXCEEDED
   and
   065 - OBJECT PROGRAM SIZE LIMIT EXCEEDED

5. Resetting the user specified time limit, the timer (.LIMTR), and the generated code bit in .LSWTH:

   These will be reset in TSSH or TSSL at the end of the interaction.

Programming Method

OBJTIM is nonreentrant.

Interrupts are not inhibited.

Storage

No internal storage is used.

Other Routines Used

Core Size Adjustment ALL7AC (EP3 of TSSL)

PASS FILE TO REMOTE BATCH PROCESSOR

PASFLR (DRL PASFLR) generates the information required by GEIN to process this job. It performs the same functions as DRL PASFIL for jobs designated REMOTE.

CALLING SEQUENCE

PASFLR is called from any TSS subsystem through EXDRL via:

```
8       16
 DRL    |PASFLR
 Return |
```

For further details, see DRL PASFIL.

SEND "STOP PAPER TAPE" TO DATANET

PPTOFF/PPTERM (DRL STOPPT) sends the command, Stop Paper Tape Input, to the DATANET 355 prior to sending an error message to the remote terminal. There are two entry points, one from the TSS Executive and one from a subsystem.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

PPTOFF is called from the Executive via:

```
 8        16
|         |
|TSX1     |PPTOFF
|Return   |(If tape running)
|  .      |   .
|  .      |   .
|Return   |(If tape not running)
```

PPTERM is called from any TSS subsystem through EXDRL via:

```
 8        16
|         |
|DRL      |STOPPT
|Return   |
|         |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

```
8       16
TSX1    RETSBS  (If no tape running)
MME     GEENDC  (If tape was running)
```

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRL STOPPT is nonreentrant.

Interrupts are not inhibited.

Storage

Internal temporary storage is used to store the entry flag.

Other Routines Used

MME GEROUT Processor GROUT (EP1 of .MROUT)
Line Service LINSRV (EP1 of TSSJ)

SAVE PROGRAMS ON PERM FILE

SAVE (DRLSAV) allows any program to be saved on a perm file instead of requiring a perm file to be built via a batch activity. A maximum of 40 files may be contained in a single catalog.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

SAVE is called from any TSS subsystem through EXDRL via:

```
8         16
DRL       SAVE
ZERO      Name pointer, 0 or 1*
ZERO      Initial address of program to be written,
            Final address of program to be written
ZERO      Entry address, load origin
ZERO      Return address, buffer location
```

* 0 when first program being written on file.

1 when program is being added to an existing file of program(s).

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV via:

```
 8         16
|TRA       |**
|TRA       |LINSRV
|TRA       |0,1
```

POSTCALLING SEQUENCE

A - contains 0 if SAVE successful

Bit 35 = 1 in A - File not large enough for requested SAVE.

SUPPORTING INFORMATION

Programming Method

SAVE is nonreentrant and written in floatable code.

Interrupts are not inhibited.

Storage

No internal storage is used.

Other Routines Used

Search the AFT SRCH (EP8 of TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
Executive Disc I/O TSXDIO (EP3 of TSSK)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
End Courtesy Call ENCC (EP5 of .MDISP)
Print Error Message PNTERR (EP1 of TSSG)
Special Entry PNTTSX (EP2 of TSSG)

SPAWN A SPECIAL BATCH ACTIVITY

TASK (DRL TASK) provides entry to the batch system at the core allocation level. Since peripheral allocation is bypassed, all file PAT's must be provided in the SSA information passed via *J.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TASK is called from any TSS subsystem through EXDRL via:

```
8         16
LDA       0,DL
DRL       TASK
ZERO      SSA buffer, location of file list
Return
```

OPERATING SYSTEM INTERACTION

The top entry in the stack is the IC and I of the calling routine.

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

```
8         16
TSX1      RETSBS
```

POSTCALLING SEQUENCE

A-register contains one of the following return codes:

0 No error (DRL) - batch status in Q-register
1 Undefined file
2 No SNUMB
3 Duplicate SNUMB
4 No program number available
5 Activity name undefined
6 Illegal user limits (time, size, etc.)
7 Bad status (R/W *J)
8 System out of file space for pushdown file
9 No *J provided

SUPPORTING INFORMATION

TASK completes the information in the SSA after verifying that provided by the user. The PAT's for the designated files are copied to the SSA buffer. The SSA is then written on the first two blocks of the user's *J. GEPOP is enabled to enter the job at the core allocation level (bypass GEIN, PALC, etc.). The calling SSA is put in wait status. GECOS will notify TSS when the batch activity is completed.

Upon return from the GECOS activity, TASK is again entered. The IC and I is not modified during the first phase and a unique value is stored in the user's A-register to denote the second entry to TASK. The first two blocks of *J are read into the user's SSA buffer. This is the SSA upon completion of the batch activity. The file list designates those file PAT's which must be updated in the user's AFT (in case of file grow, etc.). The status of the batch activity is returned in Q-register.

In the event a user "breaks" or "disconnects" his terminal while the spawned job is executing, the break or disconnect must be delayed until the DRL TASK has completed the second phase.

Programming Method

TASK is nonreentrant and written in floatable code.

Interrupts are not inhibited.

Storage

No internal storage is used.

Other Routines Used

Provide Date and Time of Day FGTIM (EP8 of .MFALT)
Search the AFT SRCH (EP8 of TSSK)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Process Specific Device Request ENTRY (EP1 of .MALC2)
Executive Disc I/O TSXDIO (EP3 of TSSK)
Update Allocator Queue ALLOC (EP1 of TSSL)
Line Service LINSRV (EP1 of TSSJ)
Relinquish RLC (EP4 of .MDISP)
Enable Program ENB (EP6 of .MDISP)
End Courtesy Call ENCC (EP5 of .MDISP)
File Spacing TSXSPF (EP7 of TSSK)
Return Links/Llinks to the Pool ONE (EP1 of .MALC9)
Delinking a File FREE (EP3 of TSSD)
File Entry Space Allocation PATMAN (EP1 of TSSD)
Print Error Message PNTERR (EP1 of TSSG)

SIMULATED KEYBOARD INPUT

FAKEIN (DRL PSEUDO, TSSK) allows one subsystem to store data in the UST I/O buffer so the data can be recovered by another subsystem via a DRL KIN. The effect is to create a 62-word core file in the UST.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

FAKEIN is called from any TSS subsystem through EXDRL via:

```
 8       16
 DRL     PSEUDO
 ZERO    Location of Tally, Status
 Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

```
 8       16
 TSX1    RETSBS
```

Error return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

A subsystem "X" would use DRL PSEUDO to pass "input" to a subsystem "Y" that would be activated by a DRL CALLSS. Eventually, Y will return to X, and the process may be continued. Subsystem X thus becomes a 'user surrogate' subsystem with TSS subsystems available to it as a "library."

No check on the actual data placed in the UST will be made, therefore, any type of data may be passed in any form - binary, USASCII, etc. DRL KIN also takes no note of the data form.

The tally is used by the derail processor in transferring the data (in 9-bit characters) and must therefore be reset prior to each subsequent use.

The status return word is provided for future expansion and will not be used in the initial implementation.

No more than 62 words may be transferred at any time. Each successful transfer will overlay information already in the UST since buffer loading will begin at the start of the buffer.

In operation, the sequence of events is as follows -- upon receipt of the PSEUDO derail, the derail processor examines the tally to determine that (a) the tally is not run out, (b) the tally is not for 6-bit characters, (c) the tally is not set to transfer more than 248 characters (62 words), (d) neither the starting nor ending addresses specified by the tally are outside subsystem limits, and that the status word is not outside subsystem limits. Should any of these conditions occur, the subsystem will be aborted, and an error message sent to the remote terminal.

Once these checks have been successfully completed, the derail processor checks to determine whether or not any output remains in the UST buffer and flushes any that it finds. When the buffer is cleared of all output, the data to be moved is placed in the buffer and the buffer control words set up exactly as if the data had come in from a remote terminal. Transfer will end when the tally runs out.

Once all material has been moved, the derail processor returns to the second location past the DRL PSEUDO in the calling subsystem.

This derail is not available to DATANET 760 terminals.

SUPPORTING INFORMATION

Programming Method

FAKEIN is nonreentrant and written in floatable code.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for storing tallies.

Other Routines Used

Print Error Message PNTERR (EP1 of TSSG)
Buffer Dump BUFDMP (EP4 of TSSK)

ALLOCATION AND DISPATCH OF SUBSYSTEM PROGRAMS (TSSL)

TSSL selects the order of execution for subsystem programs eligible for processing, manages the available core and mass storage space, swaps the programs into and out of core, and dispatches the processor to the program.

TSSL contains four entry points:

- EP1 Update Allocator Queue (ALLOC)
- EP2 Allocate Job in Queue (ALLOCI)
- EP3 Core Size Adjustment (ALL7AC)
- EP4 Set Up for Swap (SWAP)

TSSL occupies approximately 3410 (octal) core storage locations.

A job is allowed a given amount of processor time before it is considered eligible for swapping. The assigned time is a function of the program size, with a minimum of 200 milliseconds and a maximum of 1 second. To avoid the overhead required to select a different job each time the time-sharing system is interrupted, the processor is returned to the "last" job until the given amount of time has accumulated, or an I/O request is initiated.

The Allocator queue is maintained in descending order of priority. The current priority value is stored in lower half .LBACK. When a new job is entered into the queue, the priority value is infinite since one objective is to maximize the rate of user/system interactions with emphasis on the first responses for short jobs. Long jobs gradually drop down in the queue as times accumulate, but only to a certain point.

A pass through the allocation queue is made about every 2 seconds to do the following for each entry:

- Update the priority values and reorder the queue if necessary.

- Start swap-out if keyboard I/O has been initiated and any bypass counts have accumulated.

- Increment a bypass number. If this number is greater than a given bypass count, the urgent user count is incremented and a pointer to the most urgent user is kept.

Check for any urgent users and, if any, calculate average urgency. If one most urgent user has an urgency at least twice the average, immediate action is taken to load and process that job.

Make a pass through the Allocator queue for jobs eligible to be loaded/swapped into core. Note that jobs which have maximum priority will be at the top of the queue and, therefore, will be moved into core quickly. As a job is deemed eligible, space will be found in core and the load/swap started. As many jobs as will fit are moved into the core area.

The jobs residing in core are checked. If a job has accumulated three times the allowed processor time, this job is eligible to be swapped out if a job in the Allocator queue is eligible for swap-in. Therefore, the Allocator queue is searched for an eligible job which will fit in the core space to be made available. If found, swap-out is started with a courtesy call to start the swap-in.

Now the Allocator queue is again scanned to select a job ready for the processor and having the least amount of processor time for this job. The bypass number is cleared and the UST pointer is saved as "last" job.

The core Allocator maintains a memory map of the time-sharing system core memory in a linked table consisting of 2-word entries.

word 1	Forward link pointer	No. blocks used	No. blocks hole
word 2	IA (core location) Backward index	UST Pointer	

Core blocks are 1024 words and all IA core(s) are on block boundaries. (To facilitate calculations, the "No. Blocks" in core map are two times the number of 1024-word blocks. Since base addresses are set for subsystems, block boundaries are required. Another word (.TAMPT) containing pointers to the initial entry and the last entry of the core map must also be maintained. At initialization (TSTART), the initial address mod 1024 of time-sharing system core memory will be established (.TACOR). (TSTART is last in the deck and is overlayed with time-sharing system core memory.)

When the core allocation routine is called, new/old interaction is checked. Search for space for a new job is from the low (address) end of core, that is, forward through the map. An old job causes the search to start from the high end of the core backward through the map.

When space is found, a new entry is made and hole sizes, forward pointers, and backward indexes are modified to reflect the change. When searching for space, the smallest hole large enough to meet requirements is found.

The core space is cleared (0) before loading.

There are three possible results (returns) of core allocator routine:

No core available

Not enough contiguous core available

Space found - return IA (core)

UPDATE ALLOCATOR QUEUE

ALLOC (EP1 of TSSL) updates the Allocator queue by entering changes in status of subsystem programs in execution.

The Allocator queue is made of linked UST's which have a job to be executed. They are linked both forwards and backwards, with starts and ends stored in .TSCOM. The forward link pointer is in the upper half of .LFLAG and the backward pointer in .LBACK.

PRECALLING SEQUENCE

Prior to entering ALLOC, the registers listed must contain the data indicated.

X2 Pointer to UST
A Function parameter, as follows:

0 - Enter a new job (interaction) into the queue. The link pointers are stored and the new job and Allocator queue flags set "on" in .LFLAG.

1 - Remove (kill) a job from the queue. After a delay, if in swap status, any allocated core space is released. Space on the swap file is also released if it exists. The Allocator queue pointers are updated to bypass this job and the Allocator flags in .LFLAG are cleared.

2 - Set .LFLAG to indicate start of file I/O.

3 - Set .LFLAG to indicate file I/O finished.

4 - Set .LFLAG to indicate start of keyboard I/O.

5 - Set .LFLAG to indicate keyboard I/O finished.

6 - Remove the job from core to drum and remove its entry from the core map. DRL CALLSS uses this function.

7 - Release memory (core space) for the current process according to the contents of Q. The Q-register designates the number of words low (in upper half) or high (in lower half) to be released. The numbers are multiples of 1024. Modifications are made in the core map and all common references to base registers, initial address or size. DRL RELMEM uses this function.

8 - Extend memory (core space) for the current process according to the contents of Q-lower. Memory can be extended only on the high end. If contiguous space is not available, the job is swapped to the drum. The core map is modified and all common references to size are updated. DRL ADDMEM uses this function.

9 - Set bit 25 on in user's flag word to indicate that Line Service should look for a possible disconnect. DRL SPAWN uses this function.

CALLING SEQUENCE

ALLOC is called from PASFIL, TERM, PATDEA, FREE, SCAN3, TRMCU, UPDAT, DRLDIO, CALLSS, SYSRET, RELMEM, ADDMEN, STPSYS, RESTOR, GROW, CONSOL, ERROR, DRLRET, DRLKON, DRLKOT, DRLDSC, DRLDIO, FILACT, SAVE, TASK, EXENTR, EXTSQ and NEWU76 via:

```
8         16
TSX1      ALLOC
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to caller.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

ALLOC is nonreentrant and relocatable.

Interrupts are inhibited throughout.

Storage

No internal temporary storage is used.

Other Routines Used

System Error ERROR (EP1 of TSSF)
Line Service LINSRV (EP1 of TSSJ)
Trace Recent Events TSEVNT (EP1 of TSSB)
Relinquish RLC (EP4 of .MDISP)
Return Drum Space RETSPC (EP2 of TSSC)
Set Up For Swap SWAP (EP4 of TSSL)
End Courtesy Call ENCC (EP5 of .MDISP)
Print Error Message PNTERR (EP1 of TSSG)

ALLOCATE JOB IN QUEUE

ALLOCI (EP2 of TSSL) examines the Allocator queue and selects the job to be processed. It controls a program's access to core and processor time according to the priority value maintained for each job in the queue.

PRECALLING SEQUENCE

Prior to entering ALLOCI, the registers listed must contain the data indicated.

X2 Pointer to UST

CALLING SEQUENCE

ALLOCI is called from LINE SERVICE via:

```
8         16
TSX1      ALLOCI
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to the calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

ALLOCI is nonreentrant and relocatable.

Interrupts are inhibited while modifications are made in core map, scanning queue for dispatch, and moving in/out of core.

No internal temporary storage is used.

Other Routines Used

Set Up For Swap SWAP (EP4 of TSSL)
Core Allocator ALL7AC (EP3 of TSSL)
Line Service LINSRV (EP1 of TSSJ)
Request Drum Space DRMSPC (EP1 of TSSC)
Return to Subsystem Execution RETSSX (EP3 of TSSM)
SMF GEINOS Processor INOS (EP5 of MIOS)

CORE SIZE ADJUSTMENT

ALL7AC (EP3 of TSSL) examines the request for TSS core size change from the 600 console.

1. If core size request is same as current size, returns.
2. If the request is for more core, a MME GEMORE is used to request space. If not available, ALL7AC tries for 20 times at 3-second intervals before issuing message to 600 console that space is not available.
3. If the request is for less core, ALL7AC releases the space freed up after swapping user's subsystems to swap file.

PRECALLING SEQUENCE

Prior to entering ALL7AC, A upper contains the size of core requested.

CALLING SEQUENCE

ALL7AC is called from LINSRV via:

```
       8       16
       TSX1    ALL7AC
       Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

ALL7AC is nonreentrant and relocatable.

Interrupts are inhibited while making adjustments in core map.

Storage

No internal temporary storage is used.

Other Routines Used

MME GEMORE Processor MORE (EP1 of .MMORE)
MME GEINOS Processor INOS (EP5 of .MIOS)
Request Drum Space DRMSPC (EP1 of TSSC)
Set Up for Swap SWAP (EP4 of TSSL)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Memory Release for Time-Sharing GMRLM (EP9 of .MFLT1)

SET UP FOR SWAP

SWAP (EP4 of TSSL) swaps jobs in and out of core according to their priority.

PRECALLING SEQUENCE

Prior to entering SWAP, the registers listed must contain the data indicated.

A	Read/Write command
Q	Core location, courtesy call location
X2	UST address

CALLING SEQUENCE

SWAP is called from DYUST via:

```
 8       16
|TSX1   |SWAP
|Return |
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

SWAP is nonreentrant and relocatable.

Interrupts are inhibited throughout except at MME GEINOS.

Storage

No internal temporary storage is used.

Other Routines Used

Monitor Procedures MONCK (EP6 of TSSJ)
MME GEROUT Processor GROUT (EP1 of .MROUT)
MME GEINOS Processor INOS (EP5 of .MIOS)
System Error ERROR (EP1 of TSSF)

EXECUTIVE SUBPROGRAM (TSSM)

The Executive Subprogram (TSSM) is the entry from GECOS III for all faults while in execution mode and the entry for all faults and interrupts subsequent to a GELBAR setting in subsystem mode. TSSM also contains the common subroutine used to enter a subsystem and the routine used by Line Service to process entries in the time-sharing system queue.

TSSM contains three entry points:

- EP1 Entry From GECOS (EXENTR)
- EP2 Process TSS Queue Entries (EXTSQ)
- EP3 Return to Subsystem Execution (RETSSX or RETSBS)

TSSM occupies approximately 1220 (octal) core storage locations.

ENTRY FROM GECOS

EXENTR (EP1 of TSSM) calculates accounting charges, examines the fault type, and transfers control as necessary to continue processing.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

EXENTR is called from .MDISP.

Normal entry is from GECOS III; the user has no control of entries.

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Control passes - no return:

EXDRL - process DRL fault (user)
LINSRV - next task
MME GELBAR - dispatch
PNTERR - user fault
ERROR - fatal user fault

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

TSTART stores transfers to EXENTR in the fault vector to handle faults occurring in executive mode. These faults are "system" faults; therefore an error code is loaded and ERROR called. Control then continues to Line Service.

When a return from a MME GELBAR occurs, control enters EXENTR via the instruction stored at decimal location 19 by TSTART. After the registers are saved, the information at location 25 is examined to determine the type of return.

Note

> A GELBAR return occurs when a fault, a timer runout, or return from interrupt processing occurs when the processor is assigned to a subsystem.

If the return is from I/O interrupt processing, the GELBAR is reissued. If from a timer runout, accounting charges are calculated for the user and control goes to Line Service. Lockup, op-not-complete and memory parity are considered fatal for the user by calling ERROR and returning to Line Service. On a DRL fault, the user's accounting charges are calculated and control is sent to the derail processor (EXDRL).

For all other faults, memory, MME, divide check, etc., a test is made to determine if the subsystem has a fault vector; i.e., the subsystem is prepared to handle the fault. If so, the fault transfer location is stored in the IC and I, the fault transfer location is zeroed to prevent unintentional looping in the subsystem, and the GELBAR is reissued to continue execution of the subsystem. If the subsystem did not have a fault transfer stored, the user's file list is searched for a file named ABRT. If found, the subsystem is dumped to that file. In any case, PNTERR is called to issue an error message to the user.

Programming Method

EXENTR is nonreentrant and relocatable.

Interrupts are inhibited while SSA values (times) are being referenced.

Storage

No internal temporary storage is used.

Other Routines Used

Line Service LINSRV (EP1 of TSSJ)
Derail Entry From Executive EXDRL (EP1 of TSSK)
System Error ERROR (EP1 of TSSF)
Reset BAR to Smaller Area FLBAR (EP14 of .MFALT)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Search the AFT SRCH (EP8 of TSSK)
Update the Allocator Queue ALLOC (EP1 of TSSL)
File Spacing TSXSPF (EP7 of TSSK)
Executive Disc I/O TSXDIO (EP3 of TSSK)
Print Error Message PNTERR (EP1 of TSSG)
Special Entry PNTTSX (EP2 of TSSG)
End Courtesy Call ENCC (EP5 of .MDISP)

PROCESS TSS QUEUE ENTRIES

EXTSQ (EP2 of TSSM) checks the TSS queue for entries from the computer console or from GECOS III at termination of a time-sharing batch job. The parameters and functions are:

1 - Return Batch Job
2 - New Users will be Accepted
3 - No New Users to be Accepted
4 - Status Request
5 - Change Size of Core
6 - Issue Warning Message to All Active Users
7 - Request and Issue Message to All Active Users
8 - Print Last Message on 600 Console

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

EXTSQ is called from LINSRV via:

```
8         16
TSX1      EXTSQ
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRTSQ gate is shut while loading the ID and updating the tally.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

One entry in the queue is processed per interval. The intervals at which Line Service checks are frequent enough so that this should be sufficient. Changing the TRA at the end of each function could cause all entries to be processed.

NEW, NONEW and STATUS requests simply set indicators which Line Service recognizes. Size n stores the requested memory size so that the Allocator or Line Service can process. WARN t is put in the communications buffer with the given parameter "time" filled in. Flags are set for all active and new users to receive the message.

When a batch job (BATCH) is returned by GECOS, the UST must be found by means of SNUMB and the ABORTCODE stored. Then the user must be entered back in the allocation queue.

When the computer operator types MESS and the entry in the queue is processed, a request for the operator to type a message is issued: *TSS MESS--. When the operator has typed the message, it is converted to an ASCII comment and output to all current users as well as users who subsequently dial into the system.

When the computer operator types TSS LAST, the ASCII all points message is converted to BCI and output to the 600 console.

Programming Method

EXTSQ is nonreentrant and relocatable.

Interrupts are inhibited while opening/shutting .CRTSQ gate.

Storage

No internal temporary storage is used.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
End Courtesy Call ENCC (EP5 of .MDISP)
MME GEINOS Processor INOS (EP5 of .MIOS)

RETURN TO SUBSYSTEM EXECUTION

RETSSX (EP3 of TSSM) constructs the current base for the subsystem and issues a MME GELBAR.

Note:

> RETSBS is also an internal symbol of EXDRL(TSSK) for a routine checking all returns to subsystem execution. To avoid any conflict with TSSM - RETSBS, the symbol RETSSX is synonymous and used by EXDRL(TSSK) when the return to subsystem execution is valid.

PRECALLING SEQUENCE

Prior to entering RETSSX, the registers listed must contain the data indicated.

```
X2   UST address
 A = 0, subsystem entry point
   or
 A = Increment value,0
```

CALLING SEQUENCE

RETSSX is called from EXDRL or ALLOC via:

```
 8        16
|TRA     |RETSSX
| or     |
|TSX1    |RETSBS
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is via a MME GELBAR.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

The first entry to a new job is with 0, Loc in the A-register. Loc is the subsystem entry point (≠0). Thereafter, the subsystem is reentered with the n,0 in the A-register. The n (≥0) is the incremental value to be added to the IC before returning. Either n or Loc must be given but not both. The communication cell, .TCLOC, and UST entry, .LSIZE, are assumed to be set. The base register value is computed, unless it is a master subsystem in which case the TSS base is used, and the subsystem is dispatched via a MME GELBAR.

On any entry other than the first, a check is made to determine whether TSSJ detected an incoming BREAK that is to be passed to this subsystem. If this condition exists, the MME GELBAR dispatch is made via a fault vector located in cells 12 and 13 (octal) of the subsystem prefix. If the fault vector is improperly set up, TSSM exits to TSSE, which then transmits the message ILLEGAL BREAK VECTOR to the remote terminal and returns the user to system level.

The processor time spent in subsystem execution is accumulated by time-sharing system Executive for accounting purposes. Upon entry to a subsystem (RETSBS) the processor time is stored (.TEPTS). Upon return to time-sharing system Executive from the GELBAR, the processor time is stored (.TCPRT). When a subsystem loses control because of timer run out or a fault, the processor time (.TCPRT-.TEPTS) is accumulated for the user (in .LTP of UST) and the core seconds are calculated and accumulated (in .LSTP of UST).

Programming Method

RETSSX is nonreentrant and relocatable.

Interrupts are inhibited while calculating processor time.

Storage

No internal temporary storage is used.

Other Routines Used

Reset BAR to Smaller Area FLBAR (EP14 of .MFALT)
Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)

UST GENERATION (TSSN)

UST Generation routines build and release User Status Tables (UST) on an as-needed basis. At the start, the number of UST's which fit in the space before the swap core file (block boundary of core) is calculated and kept as the minimum number of UST's to be retained when space is released.

TSSN contains three entry points:

- EP1 Dynamic UST Generator (DYUST)
- EP2 UST Collect (CUST)
- EP3 Release UST (RUST)

TSSN occupies approximately 270 (octal) core storage locations.

DYNAMIC UST GENERATOR

DYUST (EP1 of TSSN) acquires space, 1k at a time, from the swap core file and builds the UST.

PRECALLING SEQUENCE

Prior to entering DYUST, the following locations in .TDCOM contain the data indicated.

```
.TCUST    - Pointers to first and last UST's in use
.TAUST    - Pointer to high address of UST's, minimum
            number to be maintained
.TAUST+1 - Amount of borrowed core (nk)
.TAUST+2 - First UST in last k of core, last UST
            used in this k of core
.TAUST+3 - UST collect flag, RELC counter
```

CALLING SEQUENCE

DYUST is called from LINSRV via:

```
8        16
TSX1     DYUST
Return   (cannot generate UST immediately)
Normal return (L(UST) in .TCUST)
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DYUST is nonreentrant and relocatable.

Interrupts are inhibited while modifications are made in core map.

Storage

No internal temporary storage is used.

Other Routines Used

Input Collector File Space Manager SYMAN) (EP7 of TSSI)
Relinquish RLC (EP4 of .MDISP)
Request Drum Space DRMSPC (EP1 of TSSC)
Set Up for Swap SWAP (EP4 of TSSL)

UST COLLECT

CUST (EP2 of TSSN) checks to see if the UST's need to be compacted, and if so, checks to see if this current user's UST is one which needs to be moved and if it can be moved at the present time. If these conditions are met, CUST moves the UST to the lowest available UST slot, adjusting the pointers within the UST, and the ALLOCATOR pointers (.LBACK and .LFLAG). CUST then transfers to clear the old UST and return space to the swap area if possible.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

CUST is called from SCAN3 via:

```
8       16
TSX1    CUST
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to LINSRV via:

```
TSXI    LINSRV
```

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

CUST is nonreentrant.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for the table of addresses to be ignored in the move.

Other Routines Used

Release UST RUST (EP3 of TSSN)
System Error ERROR (EP1 of TSSF)
Line Service LINSRV (EP1 of TSSV)
Clear UST and Make Accounting Entry TRMCU (EP2 of TSSI)

RELEASE UST

RUST (EP3 of TSSN) releases unused UST's and returns the space to the swap core file, 1 k at a time.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

RUST is called from TERMPG via:

```
8        16
TSX1     RUST
Return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to calling routine.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RUST is nonreentrant and relocatable.

Interrupts are inhibited while modifications are made to core map.

Storage

No internal temporary storage is used.

Other Routines Used

Input Collector File Space Manager SYMAN (EP7 of TSSI)

TSS STARTUP ROUTINES (TSSO)

TSSO Startup Routines (TSSO) perform the one-pass initialization of the time-sharing system. TSSO also includes the majority of the routines to handle DATANET 760 terminals. The routines are called as required by Line Service and the derail processors. They are in reality an extension of Line Service but are located in TSSO so that the code and buffers may be overlayed by UST's if no DATANET 760 terminals are to be used.

There are five functions for DATANET 760 processing as follows:

- New DATANET 760 User (GOOD76)
- Process DATANET 760 Input (760LNO)
- Reset Flags and Indicators (NEWU76)
- Release Buffer (TERM76)
- Move Output to Buffer (KOTB9)

Since these terminals are not common to all installations, users without them are not burdened with the space consuming code and buffers required for their implementation. Through a .TSCOM cell (.T760) the installation notifies TSS at startup time how many, if any, terminals will be allowed on the system at any one time. This cell can be changed whenever the GECOS III system is booted.

If no DATANET 760 terminals are allowed, MAIN (TSTART) recognizes the fact and allows the UST's to overlay TSTART, as is normally done, and also the DATANET 760 coding which will not be required.

If n DATANET 760 terminals are to be allowed, TSTART retains the coding, generates n 320-word buffers required for the operation and then initializes the UST's to start after this. TSTART is overlayed by the DATANET 760 buffers and the UST's.

The features of the DATANET 760 which must be recognized and which affect other TSS routines are:

1. Character set
2. Maximum of 48*26 characters transmitted at one time
3. Desirability of paging output

Other Routines Affected by DATANET 760

TSSA (Communication Region)

In block .TSCOM a cell is defined:

```
     .T760   ZERO  n,0     Number of DATANET 760's, 0
```

TSSH (Control of User Processes)

When the EXEC primitive is to be executed, clear the character and line counters and issue a form feed (clear screen and position cursor at top left corner).

TSSJ (Line Service)

1. As a new user enters the system, the terminal type is checked. If it is a DATANET 760, the routines in TSSO are initialized to assign a buffer, set flags and indicators, etc.

2. While processing terminates, status is tested for "PRT". If found, the pseudo-courtesy call is answered to issue a FF.

3. When a user enters build-input mode:

   Use of auto mode is inhibited.

   A special procedure (760LNO) must be followed in order to process multiple line input.

4. When errors occur, the pointer to the DATANET 760 buffer must be used for the input request.

TSSK (Derail Processors)

1. KOUT, KOUTN, KIN

   For an out/in request, affix an additional CR-LF at the end of the message. This is a convenience for the user as it facilitates the positioning of the cursor for transmission of data.

On input, determine if input pointer is to the DATANET 760 buffer or UST buffer and locate the character count accordingly. When moving data from the DATANET 760 buffer replace the final ETX with CR.

On output, examine each character for DATANET 760 control characters (FF, LF, CR, PR, etc.) and act accordingly.

2. TERM -- Deallocate the DATANET 760 buffer (TERM76).

3. NEWU -- Reset flags and indicators and initialize the buffer (NEWU76).

TSSL (Allocator)

Do not issue the null characters at the swap.

TSSN (Dynamic Ust, DYUST)

No initial UST is assembled with DYUST, as previously. Startup must first determine if DATANET 760 terminals are allowed, in which case the DATANET 760 buffers must precede the USTs.

NEW DATANET 760 USER

The GOOD76 function of TSSO locates a DATANET 760 buffer for the user and links it to th UST. Buffer locations and usage are tabulated at .TL760 in KOTB9. The line counter is initialized according to terminal type code as follows:

Type	Screen lines
5	4
6	8
7	16
10	26

.TL760 contains the following data.

```
     Word 0   Number of DATANET 760's allowed, number used
     Word 1   Buffer location, 0 or UST pointer
       .        .                    .
       .        .                    .
```

760LNO 760LNO
TSSO TSSO

PROCESS DATANET 760 INPUT

The 760LNO function of TSSO issues the input requests for DATANET-760 when in build-input mode.

Prior to entering 760LNO, the buffer contains the data indicated.

Word 0	*+4, UST pointer or zero
1	Number of lines, line counter
2	Current tally
3	Number characters per line, 0
4	Number of characters, zero
5	ASCII input

760LNO issues the input request (*) when necessary and moves the data, a line at a time, to the UST buffer for processing in the normal procedure. Whereas a MME GEROUT must be issued for every line of input from a TTY, one MME GEROUT may be issued for up to 26 lines of DATANET 760 input. In the buffer, word 0 points to the input data and the user. A zero in the right half of word 0 indicates that the buffer is available for assignment.

Word 1 indicates the number of lines allowed and the current position on the screen.

Word 2, if 0, means there is no data in the buffer. This is a signal to issue the input request (*). When data is read, the tally is established and thereafter points to the last line used.

Word 3 contains the number of characters per line for this device.

Word 4 contains the number of characters input.

RESET FLAGS AND INDICATORS

The NEWU76 function of TSSO resets the DATANET 760 flags and indicators and initializes the buffer.

Prior to entering NEWU76, the registers listed must contain the data indicated.

X2	UST address

RELEASE BUFFER

The TERM76 function of TSSO releases the DATANET 760 buffer.

Prior to entering TERM76, the registers listed must contain the data indicated.

X2	UST pointer

MOVE OUTPUT TO BUFFER

The KOTB9 function of TSSO moves the DATANET 760 output to the UST buffer.

KOTB9 is called by DRL KOUT when output for a DATANET 760 user is being moved to the UST buffer. Output paging is done here; that is, lines are counted and when the screen is full, before more output is sent, an input request is issued. When PRT status is received, an FF to clear the screen is issued and the output continued. If a normal status is received, output is continued and will overlay the previous page. (See Line Service.)

The following special characteristics and requirements are considered:

1. All alphabetic characters must be upper case.

2. No NULL characters (177) are issued.

3. Characters must be counted to determine when a line wraps to the next line for line counting purposes.

4. Control characters must be examined for their effect on the counters.

   FF - Character count and line count set to 0

   PR - Character count and line count set to 0

   LF - Increment line count by 1 or 2 depending on character count. If line count equals the screen size, dump the buffer and issue the request to continue output.

   CR - Clear character counter after testing for line wrap, in which case increment the line counter by 1

   BS - Reduce character count by 1

   RLF- Reduce line count by 1

   FS - Increment the character counter by 1

TIME-SHARING SYSTEM STARTUP

MAIN (TSSO) initializes communication cells, the fault vector, and the PAT pointers. The available TSS Subsystems are also moved to the #P or #Q files, and the program descriptor for each subsystem is initialized.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

TSSO is called from GECOS III by the .ENTRY macro.

OPERATING SYSTEM INTERACTION

.STEMP+8 and .STEMP+9 are used for link information exchange with .MALC2, EP1.

.CRPOQ is shut while a call to .MPOP6, EP2 is put into the queue.

ROUTINE RETURNS

Return is to LINSRV.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Loading the time-sharing system activates this module.

MAIN (TSTART) initializes the fault vector and clears all unused cells at addresses less than 100. The PAT pointers for SY** and other system temporary files are constructed. The subsystems are collected and stored on #P and #Q files for rapid accessibility. Normally, #P will be assigned on the fastest available device and #Q on the next available device. By means of a parameter in the Program Descriptor Name table at IN900, systems are assigned to #P or #Q. Little used systems may be assigned to #Q. Since the total required space on #P and #Q may vary from installation to installation, it is not necessary to open the files with the total required space. As more space is needed, a GEMORE request is issued.

The size of available core for subsystem execution is calculated and the core map initialized. The space occupied by TSTART module is also included as available core space.

After initializing time cells in the Communication Region, control is sent to Line Service to activate the time-sharing system.

The maximum number of time-sharing system users is specified right-justified in word .TFMAX in the Communication Region. This word is used by TSTART to determine how many slave service areas (SSA) are needed to handle the number of users specified. The number of SSA's required for n users is determined from a table in TSTART called USERS. The system as presently implemented has .TFMAX set to 18 users. The allocation of SSA's is based on the table in the following manner:

```
USERS  DEC  19      1-18 Users  Requires 2 SSA's
   +1  DEC  37     19-36 Users  Requires 4 SSA's
   +2  DEC  55     37-54 Users  Requires 5 SSA's
   +3  DEC  73     55-72 Users  Requires 6 SSA's
   +4  DEC 180   over 72 Users  Requires 7 SSA's
```

If a site's requirements are for 19 or more users, it is advantageous that the actual maximum number of users anticipated be placed in .TFMAX. The utilization of space within the slave service areas is based on the number of users specified in .TFMAX. The first slave service area is required by GECOS III and the second is used for I/O queues required to handle the number of users specified in .TFMAX. The remainder of the second slave service area plus the other slave service areas is used to contain as many PAT tables as possible (i.e., as many open files as possible). Thus by specifying the actual maximum number of users in .TFMAX, a significant gain may be realized in PAT space.

Cell .TFMAX can be changed by patching the GECOS III startup deck if the maximum number of users at a site is changed from day-to-day. .TFMAX can also be changed by reassembling the Communication Region with a new value in .TFMAX.

With the system as released, the limit on the number of files that can be open at the same time is 640.

Programming Method

MAIN is nonreentrant and relocatable.

Interrupts are inhibited while major initialization is in process.

Storage

MAIN is overlayed into TSSA by the loader and startup takes place there. TSSA is then reinitialized to its required state at conclusion of startup activities.

Other Routines Used

Let Slave Program Enter Master Mode FEMM (EP10 of .MFALT)
Relinquish RLC (EP4 of .MDISP)
Enable Program ENB (EP6 of .MDISP)
Relinquish Control Until Program Enabled DSCNT (EP11 of .MDISP)
Process Specific Device Request ENTRY (EP1 of .MALC2)
File System Initialization and Module Selection FSINIT (EP1 of .MFS09)
Roadblock GRD (EP2 of .MDISP)
Assign An I/O Entry QUEUE (EP4 of .MIOS)
Master Message Processor ITYM (EP7 of .MIOS)
Terminate Error Entry FALT (EP3 of .MBRT1)
User GECALL Load Processing MCAL (EP1 of .MCAL1)
MME GEINOS Processor INOS (EP5 of .MIOS)
MME GEMORE Processor MORE (EP1 of .MMORE)
Provide Date and Time of Day FGTIM (EP8 of .MFALT)

APPENDIX A. SYSTEM MACROS

The time-sharing system modules and subsystems require many definitions of the Communication Region, User Status Table (UST) contents, derails, etc. These have been formulated into a set of macros. They are loaded by

```
LODM  .G3TSn
```

(The proper choice of n will be defined at the time of an SDL distribution.)

The macros and their uses are as follows:

.TSCOM Should appear in all time-sharing system decks and contains all the common communication definitions and the macro call .SSUST for the UST equivalences.

.SSCOM Should appear in all master subsystem decks and contains the common communication definitions and the macro calls .SSUST and .SSDRL. These will cause the UST and DRL equivalences to be included.

.SSUST All privileged subsystems should use this macro to provide the UST and DRL equivalences.

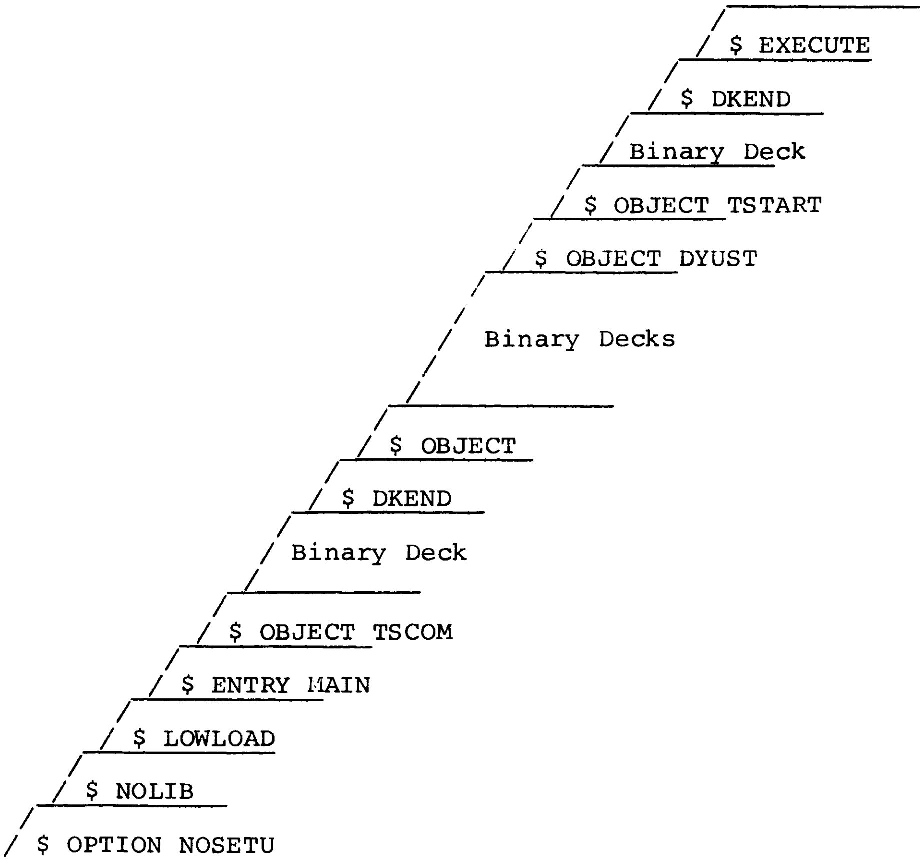
.SSDRL Normal subsystems should use this macro to obtain the DRL definitions.

PRNTTY n, (message n characters long),K

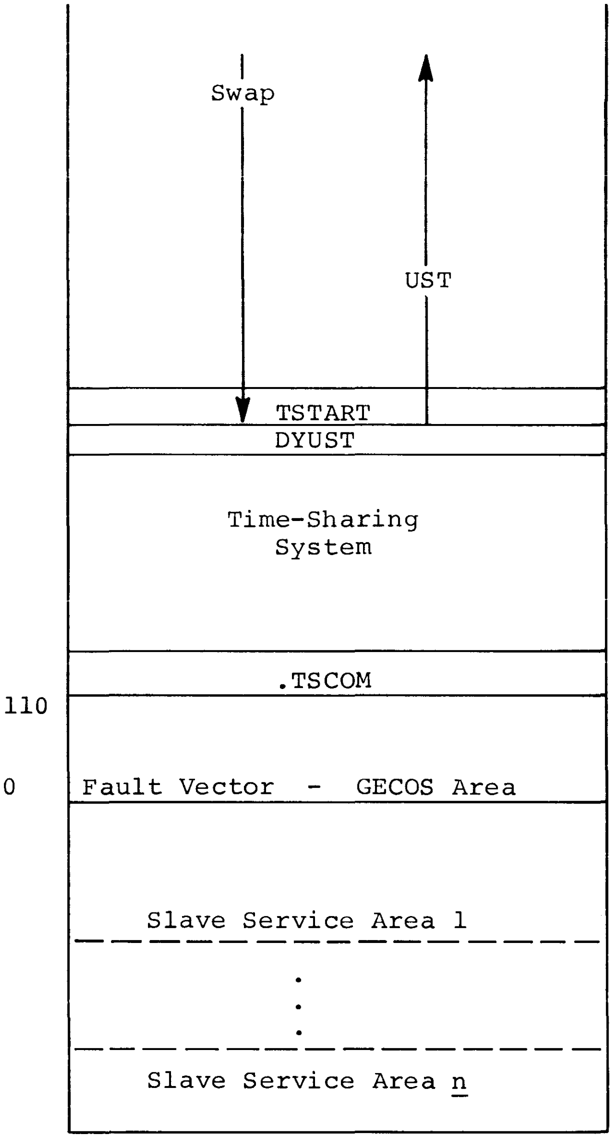
Used to print the error message from a subsystem. K (optional) is a pointer to a word containing control characters to be affixed to the end of the line (CR,LF,NULL, etc.).

APPENDIX B. TIME-SHARING SYSTEM DECK SETUP AND MEMORY LAYOUT

The deck set-up and memory layout of the time-sharing system are illustrated below. The only restriction is one that TSCOM (communication block) must be first and TSTART (TSSO) must be last. When calculating the extent of the swap area, startup assumes it can overlay itself.



Deck Setup of Time-Sharing System



Memory Layout of Time-Sharing System

The User Status Tables are constructed/released as needed. Space is obtained/released to the bottom of the swap area. Although GECOS III assigns a specific amount of core to the time-sharing system at load time, it is possible to request core size changes from the console. If efficiency or response time deteriorates because of crowded conditions in the SWAP-UST area, a message may be issued at the console to that effect.

APPENDIX C. ERROR MESSAGES

SYSTEM ERRORS

All the possible time-sharing system error messages which are output at the GE-600 operator's console are described in detail in the GE-600 Line Typewriter Messages reference manual, CPB-1477.

SUBSYSTEM ERRORS

Subsystem error messages detected by the executive and received at a terminal are listed below, with the source of the message given. The messages are number-coded when they apply to the HELP subsystem, where explanations and suggested corrective actions are given.

Error Code No.	Error Message	Source
001	INCORRECT PRIMITIVE	CONTROL
002	BAD FILE I/O COMMAND	DRL DIO
003	BAD DCW	DRL DIO
004	ADDRESS OUT OF RANGE	DRL
005	BAD DRL CODE	DRL
006	LEVEL OF CONTROL TOO DEEP	CONTROL,DRL CALLSS
007	BAD PROG. DESC.	CONTROL
008	LOOP IN PRIMITIVES	CONTROL
009	SYSTEM UNKNOWN	STARTP
010	PROGRAM TOO LARGE TO SWAP	ALLOC
011	INCORRECT CORE FILE USAGE	DRL CORFIL
012	PROGRAM NOT ALLOWED USE OF THIS I/O	DRL DIO
013	DRL ALLOWED ONLY BY LOGON	DRL USERID

014	REQUEST RELEASE OF TOO MUCH CORE	DRL RELMEM
015	CAN NOT RESET ID	DRL USERID
016	OVERFLOW FAULT	EXEC
017	ILLEGAL OP CODE	EXEC
018	MEMORY FAULT	EXEC
019	FAULT TAG FAULT	EXEC
020	DIVIDE CHECK FAULT	EXEC
021	BAD STATUS SWAP OUT #S	ALLOC
022	BAD STATUS SWAP IN #S	ALLOC
023	BAD STATUS LOAD #P	ALLOC
024	BIT POSITION > 35	CONTROL
025	YOU HAVE FILE ACCESSED WITHOUT WRITE PERMISSION	DRL DIO
026	YOU HAVE FILE ACCESSED WITHOUT READ PERMISSION	DRL DIO
027	ADDRESS IS OUTSIDE OF FILE LIMITS	DRL DIO
028	READ LINKED FILES ONLY WITH THIS COMMAND	DRL FILSP
029	ILLEGAL SYSTEM SELECTION	CONTROL
134	INVALID FUNCTION NUMBER IN DRL FILACT	DRL FILACT
135	USER CANNOT ACCESS THE SYSTEM MASTER CATALOG	DRL FILACT
138	TAP* FILE UNDEFINED	DRL DRLPT of LINSRV
139	ERROR IN WRITING TAP* TAPE	LINSRV
	DRL ABORT - FILE NOT SAVED	DRL ABORT
	DRL ABORT - PROGRAM DUMPED	DRL ABORT
	NOT ENOUGH CORE TO RUN JOB	ALLOC

	SORRY - OUT OF SWAP SPACE. TRY AGAIN	ALLOC
	FILE ADDRESS ERROR-RETRY	DRL DIO
	DRL ABORT - ERROR ON PROGRAM DUMP	DRL ABORT
	DRL ABORT - ABRT FILE TOO SMALL FOR ENTIRE DUMP	DRL ABORT
	BAD STATUS - DRL RESTOR FILE	DRL RESTOR
	RESTORE FILENAME NOT IN AFT	DRL RESTOR
064	EXECUTE TIME LIMIT EXCEEDED	EXEC
	ILLEGAL BREAK VECTOR	EXEC
065	OBJECT PROGRAM SIZE LIMIT EXCEEDED	ALLOC
	INCORRECT ENTRY TO DRL TASK	DRL TASK
	RESTOR PROGRAM NAME UNDEFINED	
	REACHED CATALOG LIMIT ON DRL SAVE	DRL SAVE
	DRL PSEUDO: TALLY INCORRECT	DRL PSEUDO
	BAD DRL SAVE DATA LOCATION	DRL SAVE
	DRL SAVE-DO FIRST PROGRAM SAVE BEFORE ADDED PROG.	DRL SAVE
	DRL SAVE ON RANDOM FILE ONLY	DRL SAVE
	TERMINAL TYPE NOT ALLOWED DRL PSEUDO	DRL PSEUDO

APPENDIX D. USER ACCOUNTING

It is necessary to accumulate usage data on an "as-you-go" basis, so that each user is charged according to his use of the system. Within the time-sharing system, four factors are considered: core-seconds, number of disc I/O requests, number of characters output to the remote terminal, and elapsed time during which the user is connected to the time-sharing system.

The data is accumulated in several locations within the system. As the times or counts are detected, they are accumulated into the User Status Table. When the user enters the system and acquires a User Status Table, the sign-on time is stored in .LTALC by LINSRV (Line Service). The difference between .LTALC and the sign-off time provides the elapsed-time factor.

In EXEC, when control is given to a subsystem, the processor time is noted. When that subsystem subsequently loses control, the processor time is multiplied by the subsystem core size and accumulated into .LSTP for the designated user. A flat charge of 2 milliseconds per thousand of core is charged for a derail (TSDRL).

In TSDRL, each request for a file I/O is counted in the lower half of .LSTIO. Also, whenever a buffer of key output is ready, the count of key characters divided by 8 is added into the upper half of .LSTIO.

The user charges are calculated in the subsystems LOGOFF, NEWU, and TERM. In LOGOFF, the calculation is made to print the signoff message for the user's information. In TERM, the calculation is made in order to update the user's information. In TERM, the calculation is made in order to update the user's accounting file. In NEWU, the calculation is made both for the message and to update the user's accounting file.

When the user is disconnected, a record is written during the termination process on the GECOS III accounting file for the user. The following is the format of the information written.

	Number of words (9 or 11)	7	Size and type record
1	.CRSID		System identification
2	.CRDAT		Date
3	ID (char. 1-6)		User ID
4	ID (char. 7-12)		
5	current time		Time of termination
6	.LTALC,2		Time user entered TSS
7	.LSTIO,2		Number of TTY char/8, Number of disc I/O
8	.LSTP,2		Core seconds used
9	Cost		Calculated Charge (¢)
10	1-6 BCI-char. chg. no.		Conditional upon use
11	7-12 BCI-char. chg. no.		of charge number

The following formula describes the computation:

Cost of time-sharing system session =

$$\Big[(.LSTP*P_1 + .LSTIO_{LOW} *.3*P_2 + .LSTIO_{UP} *P_3/C + \\ (.LTALC - Time) * P_4)\ K_5\Big]\ P_f$$

where:

$.LSTP$ = accumulated core-seconds used

$.LSTIO_{LOW}$ = number of disc I/O's performed

$.LSTIO_{UP}$ = number of characters (key IO/8)

.LTALC-Time= elapsed time

P_1 = charge for core seconds/hour

P_2 = charge for disc I/O time/hour

P_3 = charge for page of key I/O

C = average number characters/page

P_4 = charge for elapsed time/hour

K_5 = multiplication factor

P_f = priority factor

The parameters are assembled in the programs LOGOFF, TERM, and NEWU at a specific place and in a specific order to facilitate changes by assembly or patching. In each case, they will be located just following the required 100 word subsystem date area at assembled location 144 (octal) (loaded location 254 (octal)).

The parameter order, format, and present values are as follows:

DEC	600	Elapsed time price/hour	$ 6.00
DEC	1000	Core second price/hour	10.00
DEC	2400	Disc I/O price/hour	24.00
DEC	2	Price/page key I/O	0.02
DEC	1	Multiplication factor K	
DEC	100	Priority factor table	
DEC	110		
DEC	120		
DEC	130		
DEC	140		
DEC	150		
DEC	160		
DEC	170		
DEC	180		
DEC	190		
DEC	2000	Average number characters of key I/O per page	

I N D E X

Honeywell

HONEYWELL
TECHNICAL PUBLICATIONS REMARKS FORM *

TITLE: SERIES 600 GECOS III
TIME-SHARING SYSTEM EXECUTIVE

DATED: FEBRUARY, 1971

FILE NO: CPB-1501B

ERRORS NOTED IN PUBLICATION:

Fold

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

Fold

(Please Print)

FROM: NAME ______________________ DATE __________

COMPANY ______________________

TITLE ______________________

ADDRESS ______________________

____

* Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here □.

FIRST CLASS
PERMIT NO. 39531
NEWTON HIGHLANDS
MASS.

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

POSTAGE WILL BE PAID BY

HONEYWELL
151 NEEDHAM STREET
NEWTON HIGHLANDS, MASS. 02161

ATT'N: MARKETING INFORMATION SERVICES, MS 251

Honeywell

The Other Computer Company:

Honeywell

HONEYWELL INFORMATION SYSTEMS

1468
1.5271
Printed in U.S.A.