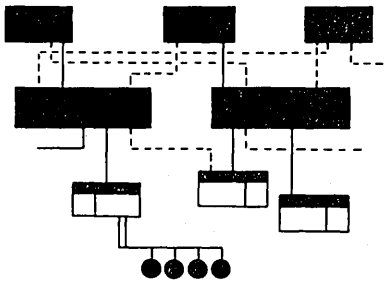
GE-625/635 GECOS-III Dispatcher and Peripheral Allocation

Information Systems

Information Systems Equipment

SOFTWARE MAINTENANCE DOCUMENT



GENERAL ELECTRIC

CPB-1491

CPB-1491

GE-625/635 GECOS-III Dispatcher and Peripheral Allocation

SOFTWARE MAINTENANCE DOCUMENT

March 1968

INFORMATION SYSTEMS

GENERAL ELECTRIC

PREFACE

This manual describes the implementation of the Dispatcher and Peripheral Allocation modules for the GE-625/635 Comprehensive Operating Supervisor (GECOS).

Additional software maintenance documents are as follows:

GE-625/635 GECOS-III Introduction and System Tables, CPB-1488

GE-625/635 GECOS-III Startup, CPB-1489

GE-625/635 GECOS-III System Input, CPB-1490

GE-625-635 GECOS-III Rollcall, Core Allocation, Operator Interface, CPB-1492

GE-625/635 GECOS-III Fault Processing and Service MME's, CPB-1493

GE-625/635 GECOS-III I/O Supervision, CPB-1494

GE-625/635 GECOS-III Error Processing, CPB-1495

GE-625/635 GECOS-III Termination and System Output, CPB-1496

GE-625/635 GECOS-III File System Maintenance, CPB-1497

GE-625/635 GECOS-III Utility Routines, CPB-1498

GE-625/635 GECOS-III Comprehensive Index and Glossary, CPB-1499

GE-625/635 GECOS-III Flowcharts, CPB-1500

GE-625/635 GECOS-III Time-Sharing System, CPB-1501

This manual was produced using the General Electric Remote Access Editing System (RAES). RAES is a time-shared disc-resident storage and retrieval system with text-editing and manuscript formatting capabilities. The contents of the manual were entered into RAES from a remote terminal keyboard, edited using the system editing language, and formatted by RAES on reproduction masters.

The index was produced using a computer-assisted remote access indexing system. This system produces an index using source strings delimited at manuscript input time.

Suggestions and criticisms relative to form, content, purpose, or use of this manual are invited. Comments may be sent on the Document Review Sheet in the back of this manual or may be addressed directly to Documentation Standards and Publications, C-78, Processor Equipment Department, General Electric Company, 13430 North Black Canyon Highway, Phoenix, Arizona 85029.

(1M 5-68)

CONTENTS

1. INTRODUCTION TO DISPATCHER

Processor dispatching is accomplished by one module .MDISP which contains the following entry point (EP) routines:

- DSP (EP1) Redispatch After Interrupt or Timer Runout
- GRD (EP2) Roadblock
- FRLC (EP3) Forced Relinquish
- RLC (EP4) Relinquish
- ENCC (EP5) End Courtesy Call
- ENB (EP6) Enable Program
- DSPQH (EP7) Program No. At Front of Queue
- DSPOT (EP8) Program No. At End of Queue
- DACNB (EP9) Accumulate Processor Time
- DRSTR (EP10) Restore State After Swap, Move, GEPR
- DSCNT (EP11) Relinquish Control Until Program Enabled
- DMIOA (EP12) Do Disc I/O Using System I/O Queue
- SCK (EP13) Set Alarm
- DSPQM (EP14) Program No. in Queue Following Interrupt
- TRACF (EP15) Save Trace Buffer and Write Out

During startup, the Dispatcher is initialized by:

- .IDISP Dispatcher Initialization

These routines are described in Chapter 2. In addition to the above routines, the Dispatcher contains various macro subroutines not logically a part of the dispatching function but located in the Dispatcher module:

- HCL .CALL macro
- HGT .COTO macro
- HEX .EXIT macro

These macros are described in Chapter 3.

A system trace routine, described in Chapter 3,

- TRACE System Trace

facilitates analysis of system dumps and is assembled as part of the Dispatcher.

To preserve the integrity of certain tables and to prevent more than one processor executing certain sections of GECOS, the following system gate subroutines are implemented in the Dispatcher:

- OPGAT Open System Gates
- SHUG Shut System Gates

These subroutines are described in Chapter 3.

A glossary and an index are included for user convenience.

2. DISPATCHING

DISPATCHER MODULE (.MDISP)

Dispatching processors to programs which can make effective and immediate use of the processor and/or peripheral subsystems (through the I/O Supervisor) is accomplished by the Dispatcher. A program, whether it is a GECOS module or a slave program, can be selected for execution only by the Dispatcher. Rules for selection are as follows:

1. If the first program in the Dispatcher queue was in a courtesy call, dispatch to that program.

2. Otherwise, examine the courtesy call queue and find the first new courtesy call that may be paid by this processor. If one can be found and the request can be satisified (see below), dispatch to that program.

3. If no new courtesy call is found, examine the Dispatcher queue. If a program is found in the Dispatcher queue, dispatch to that program.

4. Otherwise, go into the Dispatcher wait routine, where the processor is disabled. The processor will remain disabled until the next IOC interrupt or the next Connect IOC (CIOC) address to that processor. At that time, the processor attempts to find a program to dispatch to by passing through the Dispatcher queue again.

Once a program is selected, the Dispatcher checks to see that processor time remains for the program. If no time remains, the program is aborted (through the Termination module, .MBRT). Next, the Dispatcher checks to see if an Abort, GEPR, or Swap/Move request is outstanding for this program. If no request is outstanding, the program is dispatched to normally. If a request for Abort, GEPR, or Swap/Move is present, the request is checked to determine whether it may be satisified, and the program is dispatched to normally. If the request can be satisfied, the Dispatcher saves the state of the program and transfers control to the appropriate GECOS module where the required action (abort, etc) is taken.

SATISFYING A NEW COURTESY CALL REQUEST

Various conditions determine whether a request for a new courtesy call can be satisfied. A new courtesy call may be started for a program if:

1. The program is not already executing in another processor.

2. The SSA of the program is not being loaded.

3. GEPR is not in control.

4. Abort is not requested.

5. Swap/Move is not in control.

6. A courtesy call is not in control.

7. The program is not in the GEPOP queue waiting for more memory or for memory release.

8. SYSOUT is not writing for this program. If SYSOUT is writing for this program, only courtesy calls belonging to SYSOUT may be paid. (Thus, two MME GESYOTS are not started for the same program at the same time.)

9. The I/O queue holding the courtesy call does not have the Stop I/O bit on.

SATISFYING AN ABORT REQUEST

Two methods are available to request an Abort. First, an operator may have requested, through the control console typewriter, a TERM or KILL. Second, GEPR may have requested a program abort. Both methods result in turning on the Abort flag in the .STATE word of the program. Any other abort will not turn the abort flag on because the program transfers to the abort routine, FALT (EP3 of .MBRT1) itself.

A request for an Abort is honored under the following conditions:

1. The program is not already executing in another processor.

2. The SSA of the program is not being loaded and no SSA is pushed down.

3. No IC and I entry for the program is pushed down in the IC and I stack.

4. The IC and I is in slave mode unless the urgency of the program is zero.

5. The program is not waiting for GEPR.

6. GEPR, Abort, Swap/Move are not in control.

7. SYSOUT is not writing.

SATISFYING A SWAP/MOVE REQUEST

A request to Swap or Move a program is made by the .MPOPM module when memory compaction or swapping is to be done. This request is honored under the following conditions:

1. The program is not already executing in another processor.
2. The SSA of the program is not being loaded.
3. No IC and I entry for the program is pushed down in the IC and I stack.
4. The IC and I is in slave mode unless the urgency of the program is zero.
5. The program is not waiting for GEPR.
6. GEPR and Abort are not in control.
7. SYSOUT is not writing.

SATISFYING A GEPR REQUEST

A request to start GEPR is honored under the following conditions:

1. The program is not already executing in another processor.
2. GEPR is not already in control.
3. The SSA is not being loaded.
4. SYSOUT is not writing. If SYSOUT is writing, a GEPR request may only be honored if the I/O requiring GEPR action is a SYSOUT-generated I/O request.

REDISPATCH AFTER INTERRUPT OR TIMER RUNOUT

DSP (EP1 of .MDISP) sends control to the next program that can use the processor after an interrupt, relinquish, roadblock, or timer runout has occurred.

PRECALLING SEQUENCE

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

X5 LAL for program
X6 Program number
X7 Processor number

CALLING SEQUENCE

DSP is called from the Interrupt Handler (IOTRM) in the .MIOS module and from various Dispatcher routines.

```
          8           16

          .GOTO       .MDISP,1
          return
```

OPERATING SYSTEM INTERACTION

When control is given to DSP, it is assumed that the status of any program already in execution has been saved. If a program is in execution when this entry is reached, it is not ordinarily added to the queue of programs in .CRPRQ. However if a request is outstanding for GEPR, Abort, or Move/Swap and Move/Swap is not already in control, the program is put in the .CRPRQ so these requests may be satisfied. (See Chapter 1 for the conditions under which these requests are satisfied.)

ROUTINE RETURNS

The DSP routine transfers to .SICI within the program SSA. The .SICI cell contains the final code to reload the program registers and a return to the program. This code is as follows:

```
                    8           16

     .SICI          LDA         .SSA,I
                    CANA        1,DL
                    TZE         3,IC
                    LREG        .SSA+1,AD
                    RET         .SSA,I
                    LREG        32,5
                    RET         .SSA,I
```

The absolute addresses .SSA and .SSA+1 are calculated and stored in the above code by various core allocation modules.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DSP is reentrant and written in floatable code.

Interrupts are normally inhibited.

Storage

No internal temporary storage is used.

DSP occupies approximately 385 core storage locations.

Other Routines Used

Enable Program ENB (EP6 of .MDISP)
Set Alarm SCK (EP13 of .MDISP)
Program No. At End of Queue DSPQT (EP8 of .MDISP)
System Trace TRACE (.MDISP)
Dispatch Entry for Abort DISP (EP1 of .MBRT1)
Terminate Error Entry FALT (EP3 of .MBRT1)
GEPR Executive GEPRE (EP1 of .MGEPR)
Request Job Swap SWAP (EP1 of .MSWAP)

Flowchart

See CPB-1500 for the flowchart of DSP (EP1), .MDISP module.

GRD (EP2)
. MDISP

ROADBLOCK

GRD (EP2 of .MDISP) performs a roadblock on the program using it. When a program is roadblocked, it will not be a candidate for execution again until all I/O requests made by that program are satisfied. In roadblocking a program, GRD sets the Roadblock flag in the program status word.

PRECALLING SEQUENCE

All registers are saved by the macro .GROAD when used, and by the Fault Processor (.MFALT) when MME GEROAD is used.

The calling program must not be in a courtesy call since roadblocks within courtesy calls are illegal.

CALLING SEQUENCE

GRD is called from any GECOS program by:

```
8          16

.GROAD
```

or by a slave program:

```
8          16

MME        GEROAD
```

OPERATING SYSTEM INTERACTION

Processor time accumulated for the program requesting a roadblock is determined by DACNB (EP9 of .MDISP). Once all I/O requests are satisfied or if no I/O requests are present for the program, the DSPQT (EP8 of .MDISP) routine is used to put the program at the end of the Dispatcher queue. If the program has a request present for Swap/Move, GEPR, or Abort, the Roadblock flag is still set by GRD. The program is then put into the Dispatcher queue so that the Swap/Move, GEPR, or Abort action may occur. However, since the Roadblock flag is set, the program will be dispatched to only to begin Swap/Move, GEPR, or Abort.

ROUTINE RETURNS

Return is to the next instruction following the call after all outstanding I/O is completed.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

GRD is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

GRD occupies approximately 30 core storage locations.

Other Routines Used

Redispatch After Interrupt or Timer Runout DSP (EP1 of .MDISP)
Program No. At End of Queue DSPQT (EP8 of .MDISP)
Accumulate Processor Time DACNB (EP9 of .MDISP)
Terminate Error Entry FALT (EP3 of .MBRT1)

Flowchart

See CPB-1500 for the flowchart of GRD (EP2), .MDISP module.

FORCED RELINQUISH

FRLC (EP3 of .MDISP) provides a forced relinquish for the calling program.

PRECALLING SEQUENCE

Prior to entering FRLC, all registers must be saved.

CALLING SEQUENCE

FRLC is called from various GECOS programs.

```
        8          16
┌──────────┬──────────
│.CALL     │.MDISP,3
└──────────┴──────────
```

OPERATING SYSTEM INTERACTION

Processor time that is accumulated for the program is determined by DACNB (EP9 of .MDISP).

ROUTINE RETURNS

The IC and I is set to the location to which control is to be returned.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

FRLC is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

FRLC occupies approximately 25 core storage locations.

Other Routines Used

Program No. At End of Queue DSPQT (EP8 of .MDISP)
Accumulate Processor Time DACNB (EP9 of .MDISP)

Flowchart

See CPB-1500 for the flowchart of FRLC (EP3), .MDISP module.

RLC (EP4)
.MDISP

RELINQUISH

RLC (EP4 of .MDISP) causes control of a program to be relinquished until such time that I/O is completed for that program.

PRECALLING SEQUENCE

All registers are saved by the macro .GRELC when used, and the Fault Processor (.MFALT) saves all registers when MME GERELC is used.

Relinquish may be done while in a courtesy call.

CALLING SEQUENCE

RLC is called from any GECOS program by:

```
8          16
|.GRELC    |
```

or by a slave program:

```
8          16
|MME       |GERELC
```

OPERATING SYSTEM INTERACTION

Processor time accumulated for the program requesting a relinquish is calculated by DACNB (EP9 of .MDISP). When I/O is completed successfully, the status return portion of .MIOS (STRET) will test the program state word for a relinquished condition and if present will remove the condition. Then, STRET will place the program in the Dispatcher queue following any jobs requesting courtesy calls by calling DSPQM (EP14 of .MDISP).

ROUTINE RETURNS

Return is to the next instruction following the call after outstanding I/O is completed. If no I/O was outstanding at the time of the call, the return is done immediately.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RLC is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

RLC occupies approximately 30 core storage locations.

Other Routines Used

Redispatch After Interrupt or Timer Runout DSP (EP1 of .MDISP)
Program No. At End of Queue DSPQT (EP8 of .MDISP)
Accumulate Processor Time DACNB (EP9 of .MDISP)

Flowchart

See CPB-1500 for the flowchart of RLC (EP4), .MDISP module.

ENCC (EP5)
.MDISP

END COURTESY CALL

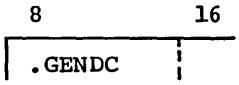
ENCC (EP5 of .MDISP) restores conditions for a program that has been in a courtesy call to those that were in effect at the time the courtesy call started.

PRECALLING SEQUENCE

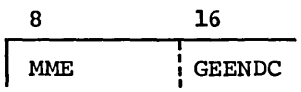
Prior to starting a courtesy call, all registers for the program are saved.

CALLING SEQUENCE

ENCC is called from any GECOS program by:



or by any Slave Program:



OPERATING SYSTEM INTERACTION

Processor time accumulated by the program in courtesy call is calculated by DACNB (EP9 of .MDISP). Program status is restored and any roadblock or relinquish condition existing at the time of the courtesy call is re-established. If the program urgency is zero or the program is processing a GEMORE or GEMREL request (dead bit is on) and no GEPR or SWAP request exists, the program is not a candidate for dispatching. Otherwise the program is placed at the end of the Dispatcher queue by calling DSPQT (EP8 of .MDISP).

ROUTINE RETURNS

Return is to DSP (EP1 of .MDISP).

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

ENCC is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

ENCC occupies approximately 50 core storage locations.

Other Routines Used

Redispatch After Interrupt or Timer Runout DSP (EP1 of .MDISP)
Program No. At End of Queue DSPQT (EP8 of .MDISP)
Accumulate Processor Time DACNB (EP9 of .MDISP)
Terminate Error Entry FALT (EP3 of .MBRT1)

Flowchart

See CPB-1500 for the flowchart of ENCC (EP5), .MDISP module.

ENABLE PROGRAM

ENB (EP6 of .MDISP) causes a program to become a candidate for execution and to be placed in the Dispatcher queue when the following conditions are met:

1. Program must be in core
2. Not already in execution
3. Not already in Dispatcher queue
4. Not roadblocked or relinquished
5. Not in control of swap routine

PRECALLING SEQUENCE

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

QR Program number in QU (bits 12-17)
Urgency in QL (bits 18-23)

CALLING SEQUENCE

ENB may be called from any GECOS-III system program.

8	16
.CALL return 0 return 1	.MDISP,6

OPERATING SYSTEM INTERACTION

System gate .CRLAL-1 is shut while interrogating the LAL table to see if the job to be enabled is known to the system and/or if it is in core. System gate entries are placed in the Dispatcher queue by routine DSPQT (EP8 of .MDISP)

ROUTINE RETURNS

Return 0 (.EXIT) Cannot Enable Program. Registers contain:

AR Retained in .STEMP8 by .CALL
QR Retained in .STEMP9 by .CALL
All index registers are destroyed.

Return 1 (.EXIT 1) Successful Enable of Program. Registers contain:

AR Retained in .STEMP8 by .CALL
QR Retained in .STEMP9 by .CALL
All index registers are destroyed.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

ENB is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

ENB occupies approximately 70 core storage locations.

Other Routines Used

Program No. At End of Queue DSPQT (EP8 of .MDISP)

Flowchart

See CPB-1500 for the flowchart of ENB (EP6), .MDISP module.

PROGRAM NO. AT FRONT OF QUEUE

DSPQH (EP7 of .MDISP) places a program in front of all others in the Dispatcher queue and sends any waiting processor to do the job.

PRECALLING SEQUENCE

Prior to entering DSPQH, the registers listed must contain the data indicated. System gate .CRDSP must be shut.

X5 LAL for Program
X6 Program number
X7 Processor number

Index registers X0 and X2 through X4 are destroyed.

CALLING SEQUENCE

DSPQH is called from the Interrupt Handler (IOTRM) in the .MIOS Module.

```
8          16
LDX1       .MDISP,DU      Get .MDISP Module No.
LXL1       .CRMOD,*1      Address of .MDISP
TSX1       7,1            EP7
```

OPERATING SYSTEM INTERACTION

When the Interrupt Handler (IOTRM) has interrupted a job that is in a courtesy call, it uses this entry to place the job ahead of all others in the Dispatcher queue.

ROUTINE RETURNS

Return is to the next instruction following the TSX1.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DSPQH is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

DSPQH occupies approximately 40 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of DSPQH (EP7), .MDISP module.

DSPOT (EP8)
.MDISP

PROGRAM NO. AT END OF QUEUE

DSPQT (EP8 of .MDISP) places a program at the end of the Dispatcher queue and sends any waiting processor to do the job.

PRECALLING SEQUENCE

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

System gate .CRDSP must be shut.

X5 LAL for program
X6 Program number
X7 Processor number

Index registers X0 and X2 through X4 are destroyed.

CALLING SEQUENCE

DSPQT is called from the Interrupt Handler (IOTRM) in the .MPOPG and several other Dispatcher routines by TSX1 DSPQT,$.

```
8          16
LDX1       .MDISP,DU
LXL1       .CRMOD,*1
TSX1       8,1
```

OPERATING SYSTEM INTERACTION

When the Interrupt Handler (IOTRM) has interrupted a program, it uses this routine to replace a job in the Dispatcher queue. When the Core Allocator (.MPOP4 and .MPOP6) has marked jobs for compaction or swapping, this routine is used.

Dispatcher routines that use this entry are:

Redispatch After Interrupt or Timer Runout DSP (EP1 of .MDISP)
Roadblock GRD (EP2 of .MDISP)
Forced Relinquish FRLC (EP3 of .MDISP)
Relinquish RLC (EP4 of .MDISP)
End Courtesy Call ENCC (EP5 of . MDISP)
Enable Program ENB (EP6 of .MDISP)
Restore State After Swap, Move, GEPR DRSTR (EP10 of .MDISP)

ROUTINE RETURNS

Return is to the next instruction following the TSX1.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DSPQT is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

DSPQT occupies approximately 40 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of DSPQT (EP8), .MDISP module.

DACNB (EP9)
.MDISP

ACCUMULATE PROCESSOR TIME

DACNB (EP9 OF .MDISP) removes a job from execution and from the Dispatcher queue and accounts for processor time used by the job going out of execution.

PRECALLING SEQUENCE

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

X5 LAL for program
X6 Program number
X7 Processor number

CALLING SEQUENCE

DACNB is called from various routines located in the .MDISP module

```
 8        16
┌──────────┬──────────
│ TSXI     ¦ DACNB,$
│ return   ¦
```

OPERATING SYSTEM INTERACTION

The current timer is compared to the last timer setting stored. The difference is added to .CRTOD in the communication region and the SSA cell .SPRT. The remaining processor time cell .SALT in the SSA is decremented by the same value.

ROUTINE RETURNS

Return is to the location following the TSXI.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DACNB is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

DACNB occupies approximately 25 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of DACNB(EP9), .MDISP module.

DRSTR (EP10)
.MDISP

RESTORE STATE AFTER SWAP, MOVE, GEPR

DRSTR (EP10 OF .MDISP) restores job conditions to those in effect before a Swap/Move or GEPR request was serviced. The program .STATE word is restored from word 7 of the current register storage area.

PRECALLING SEQUENCE

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

X5 LAL for program
X6 Program number
X7 Processor number

CALLING SEQUENCE

DRSTR is called from routines within the .MSWAP and .MGEPR modules.

8	16
.GOTO return	.MDISP,10

OPERATING SYSTEM INTERACTION

System gate .CRDSP is shut while making entries in the Dispatcher queue. Processor time accumulated by the program in Swap/Move or GEPR is calculated by calling DACNB (EP9 OF .MDISP). Any roadblock or relinquish condition that existed at the time of the Swap/Move or GEPR request is re-established. If the program urgency is zero or if the program is processing a GEMREL request and no GEPR or Swap/Move request exists, it is not a candidate for execution. The program is placed at the end of the Dispatcher queue by calling DSPQT (EP8 of .MDISP).

ROUTINE RETURNS

Return is to DSP (EP1 of .MDISP).

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DRSTR is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

DRSTR occupies approximately 25 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of DRSTR (EP10), .MDISP module.

DSCNT (EP11)
.MDISP

RELINQUISH CONTROL UNTIL PROGRAM ENABLED

DSCNT (EP11 of .MDISP) causes a job to relinquish control until it is enabled again.

PRECALLING SEQUENCE

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

X5 LAL for program
X6 Program number
X7 Processor number

CALLING SEQUENCE

DSCNT is called from routines within the .MSWAP, .MBRT1, .MGEPR, ..GEPOP, :MALC1, and .MGEIN modules.

```
8          16
.CALL      .MDISP,11
```

OPERATING SYSTEM INTERACTION

System programs that have no function to perform at the moment use DSCNT to allow themselves to be swapped out of core until such time as they are needed again.

ROUTINE RETURNS

Return is to the address plus one of the .CALL after an enable via ENB (EP6 of .MDISP) has caused the job to be placed in the Dispatcher queue and a re-dispatch takes place via DSP (EP1 of .MDISP). When control is released, however, DSP is entered immediately to attempt dispatching to other jobs in the Dispatcher queue.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DSCNT is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

DSCNT occupies approximately 5 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of DSCNT (EP11), .MDISP module.

DMIOA (EP12)
.MDISP

DO DISC I/O USING SYSTEM I/O QUEUE

DMIOA (EP12 OF .MDISP) issues I/O on disc/drum on behalf of slave programs. The routine is used for (1) swapping jobs in and out of core, (2) loading SSA modules, (3) pushing down SSA (GEPR) modules, (4) popping up SSA (GEPR) modules.

PRECALLING SEQUENCE

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

X2	Address of I/O entry skeleton
X4	Address of PAT pointer (offset to LAL)
X5	LAL for program
X6	Program number
X7	Processor number

CALLING SEQUENCE

DMIOA is called from the .MSWAP module and from various other Dispatcher routines.

```
          8          16
          |          |
          TSX1       | DMIOA,$
          |          |
or:       |          |
          LDX0       | .MDISP,DU
          LX10       | .CREMDD,*0
          TSX1       | 12, 0
          |          |
```

OPERATING SYSTEM INTERACTION

The I/O entry skeleton is moved to the System I/O Entry (.SSYIO) of the slave program. Remaining entries in the entry are filled and a seek address calculated by calling the applicable channel module (.MDR20 or .MDS20, at EP -1). The I/O is linked using LINK (EP1 of .MIOS). GEPR Override is used when issuing the I/O to allow re-issuance of I/O when an end of file fence is passed over. This is done by calling the applicable channel module at EP4. Any I/O status other than Channel Ready will be re-issued 10 times, if necessary, and then the program will be aborted.

ROUTINE RETURNS

Return is to the location following the TSX1 after successful termination of the I/O.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DMIOA is reentrant and written in floatable code.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

DMIOA occupies approximately 140 core storage locations.

Other Routines Used

Link I/O To End Of Queue LINK (EP1 of .MIOS)
Relinquish RLC (EP4 of .MDISP)
Terminate Error Entry FALT (EP3 of .MBRT1)
Negative Entry Point (EP -1 of Applicable Channel Module, .MDR20 or .MDS20)
Error and EOF Recovery (EP4 of Applicable Channel Module, .MDR20 or .MDS20)

Flowchart

See CPB-1500 for the flowchart of DMIOA (EP12), .MDISP module.

SCK (EP13)
. MDISP

SET ALARM

SCK (EP 13 OF .MDISP) sets an alarm clock for a job to ring at the end of a time increment specified by the Q-register. The job will then be enabled with urgency which is also specified in the Q-register.

QR 0 (to turn off alarm)

Time in milliseconds (bits 0-29)
Urgency of job (bits 30-35)
X5 LAL for program
X6 Program number
X7 Processor number

CALLING SEQUENCE

SCK is called from routines within the .MALC1, .MGEPR, .MGEIN, .MGEOT, and .MPOP modules.

```
8          16
.CALL      .MDISP,13
```

OPERATING SYSTEM INTERACTION

Each time DSP (EP1 of .MDISP) is entered, the alarm clock table is scanned for any jobs having an alarm set. If the specified time has passed, the alarm is turned off and the job is enabled by calling ENB (EP6 of .MDISP).

System gate .CRACK is shut while making entries in the alarm clock table.

ROUTINE RETURNS

Return is to the location following the .CALL.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

SCK is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

SCK occupies approximately 110 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of SCK (EP13), .MDISP module.

DSPQM (EP14)
.MDISP

PROGRAM NO. IN QUEUE FOLLOWING INTERRUPT

DSPQM (EP14 OF .MDISP) places a job in the Dispatcher queue following any jobs in the queue that are awaiting courtesy calls.

PRECALLING SEQUENCE

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

X5 LAL for program
X6 Program number
X7 Processor number
Index registers X0 and X2 through X4 are destroyed

System gate .CRDSP must be shut.

CALLING SEQUENCE

DSPQM is called from STRET in the .MIOS module.

```
8          16
LDX1    |  .MDISP,DU
LXL1    |  .CRMDD,*1
TSX1    |  14, 1
```

OPERATING SYSTEM INTERACTION

When a completed I/O breaks a roadblock or a relinquish condition, DSPQM allows the program to be dispatched at the earliest possible time.

ROUTINE RETURNS

Return is to the location following the TSX1.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DSPQM is reentrant and written in floatable code.

Interrupts are inhibited.

Storage

No internal temporary storage is used.

DSPQM occupies approximately 40 core storage locations.

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of DSPQM (EP14), .MDISP module.

3. SYSTEM MACRO, TRACE, AND GATE ROUTINES

This chapter describes certain routines that are not logically part of .MDISP but are assembled with it.

SYSTEM MACRO ROUTINES

System macro routines comprise the following.

- HCL .CALL macro
- HGT .GOTO macro
- HEX .EXIT macro

These are described in the following pages.

.CALL MACRO

HCL (.MDISP), the .CALL macro, manages all routine and subroutine calls within GECOS-III hard core and SSA modules. It reads SSA modules into the slave service area, pushing down the programming already there if necessary. It transfers control to those modules in hard core.

PRECALLING SEQUENCE

The .CALL macro itself does not destroy any index registers. In processing through the .CALL, the original contents of A-register and Q-register are saved in .STEMP+8 and .STEMP+9, respectively. The subroutine .CALL must recover the A- and Q-registers from those locations before it leaves the inhibited state. The .CALL does not save the index registers, and the index registers are not restored through the .EXIT routine. However, certain currently used system routines, such as LINK (EP1 of .MIOS) save registers within themselves when called. Reference must be made to the appropriate routine documentation to determine whether the registers are preserved.

CALLING SEQUENCE

The .CALL macro may be called from either an HCM or SSA module.

```
          8          16
          |          |
          .CALL      .MDXXX,Y
```

where .MDXXX is the symbolic name of any GECOS-III hard core or SSA module, and Y is any entry point number defined for the module. This macro expands in line to:

```
          8          16
          |          |
          XED        .CRCAL
          ZERO       .MDXXX,Y
```

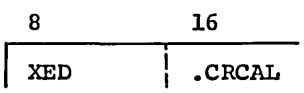
The .CALL routine obtains control through the pair of words at .CRCAL which contain the instructions:

```
 1        8          16
 |        |          |
 .CRCAL   STC1       .SSTAK+2,5*
          TRA        HCL
```

OPERATING SYSTEM INTERACTION

The information in the .CRMDD and .CRDIT tables (see CPB-1488) is used to locate any modules in core or on secondary system storage. Some modules are gated. This means that not more than one user can be executing the module at any time. The .CALL routine, by refering to the .CRMDD table and the entry word in the SSA module, enforces this rule if the module is gated. When .CALL transfers into an SSA module, it always marks the SSA module "busy."

Since the .CALL is used as a subroutine call, it always leaves the IC and I of the .CALL itself in the IC and I stack when control passes to the module called. This IC and I is removed from the stack at the .EXIT corresponding to the call itself. The IC and I is the address of the location following the



in the in-line expansion of the .CALL Macro.

ROUTINE RETURNS

Control passes to the module named at the entry point indicated by the arguments in the .CALL. All modules have an entry word containing the module number, length etc. The nth entry point is always located exactly n words after this entry word.

POSTCALLING SEQUENCE

The module called must obtain the A- and Q-registers from .STEMP+8 and .STEMP+9 before the inhibit bit is turned off.

SUPPORTING INFORMATION

If the SSA is busy when a SSA module is called, or if the length of the IC and I stack exceeds three entries, the SSA is pushed down into the pushdown file of the program before the new module is read into the SSA. This routine is part of the .CALL routine. The low order three bits of an IC and I entry are used to indicate register storage and/or SSA pushdowns. The encoding is as follows:

1. If bit 35=1, slave registers are stored in the register stack, when the IC and I entry was made.

2. If bit 34=1, a pushdown is made at the time the IC and I entry was put into the stack. The type of pushdown is indicated by bit 33.

3. If bit 33=0, all of the SSA program area was pushed down to bring in an ordinary SSA module. If bit 33=1, only the first half of the SSA program area is pushed down to bring in a GEPR module.

When a module is loaded into the SSA it is checksummed. If the checksum fails or an I/O error prevents the successful completion of the pushdown module load, ten re-tries are made. If these fail, the program is aborted. One link is assigned to each program in execution as a pushdown file. This allows six successive pushdowns. If a system error of more than this number of successive pushdowns is called for, the program is aborted. This type of abort is, of necessity, a GECOS system error not a user-caused error. The pushdown file is defined in the PAT at .SPDPA. It is also used by the system for a swap-out file in case the program must be swapped from core.

If .CALL finds the module called is already present in the SSA, it will reload the module only if the checksum is incorrect. If the module does checksum or if the checksum has been set to zero, the module will not be reloaded.

Other Routines Used

Do Disc I/O Using System I/O Queue DMIOA (EP12 of .MDISP) issues the actual commands when .CALL must push down and/or load a module from secondary storage.

Flowcharts

See CPB-1500 for the flowchart of HCL, .MDISP module.

.GOTO MACRO

HGT (.MDISP), the .GOTO macro, like .CALL, is used to transfer control between modules. It executes a transfer of control from one module to another when a return to the original module is not required.

PRECALLING SEQUENCE

The .GOTO routine does not destroy index registers. The A- and Q-registers are saved in .STEMP+8 and .STEMP+9, respectively.

CALLING SEQUENCE

The .GOTO macro may be called from either an HCM or SSA module.

```
          8           16

          .GOTO       .MDXXX,Y
```

where .MDXXX is the symbolic module name of any GECOS-III SSA or HCM module, and Y is any defined entry point to the module. This macro expands in line to:

```
          8           16

          XED         .CRGTO
          ZERO        .MDXXX,Y
          ZERO        .SMDNO,O
```

The .GOTO routine obtains control through the pair of instructions at .CRGTO which contain:

```
 1        8           16

 .CRGTO   STC1        .SSTAK+2,5*
          TRA         HGT
```

OPERATING SYSTEM INTERACTION

The .GOTO macro operates in a manner similar to .CALL except that no entry is made into the IC and I stack.

When a .GOTO occurs, the system module passes from a busy to an idle state. Thus, the .CRMDD entry for that module is marked to show the module idle. This is significant for gated modules. These are modules that may not be multi-processed because they alter contents of communication region locations without data gating. Thus, if one program is executing such a module, all others are inhibited from entering that module by the .CALL routine. When a .GOTO sets this busy bit off, another program may gain entry to the module.

ROUTINE RETURNS

Control passes to the module named at the entry point indicated by the arguments in the .GOTO. All modules have an entry word containing the module number, length, etc. The nth entry point is always located exactly n words after this entry word.

POSTCALLING SEQUENCE

The module called must obtain the A- and Q-registers from .STEMP+8 and .STEMP+9.

SUPPORTING INFORMATION

The .GOTO routine like .CALL, may push down the SSA. The discussion presented previously in .CALL is equally true here.

Other Routines Used

Do Disc I/O Using System I/O Queue DMIOA (EP12 of .MDISP) issues actual command when .GOTO must push down an SSA module.

Flowchart

See CPB-1500 for the flowchart of HGT (.MDISP).

.EXIT MACRO

HEX (.MDISP), the .EXIT macro, returns control to the IC and I at the top of the IC and I stack.

PRECALLING SEQUENCE

When an .EXIT occurs, a return is made to the location specified in the IC and I stack (.SSTAK). An entry is placed in that stack by either a .CALL or a fault (such as a MME). If the IC and I is from a .CALL, it points to the location following the XED .CRCAL. If the IC and I is from a fault, the IC and I points at the faulting cell; that is, in the case of a MME, the location containing the MME instruction. Thus, in both cases a 1 must be added to the IC address before a return can be made.

The .EXIT routine restores slave registers if they were saved in the register stack (.SREGS) at the time of the IC and I entry. If saved, bit 35 of .SSTAK entry is equal to 1. If the returning routine wishes to reset the registers before they are reloaded, the new values must be stored in .SREGS. This may be done as follows. If X4 is to be reset upon exit:

```
8           16
|-----------|-------------------------------------
| INHIB     | ON
| LDA       | .SSA,5          Get stack pointer
| LDA       | -1,AU           Get IC and I
| CANA      | 1,DL            Is bit 35 = 1
| TZE       | 3,IC            No,X4 OK
| LDA       | .SSA+1,5        Get register stack
| STX4      | -8+2,AU         Store it
| INHIB     | OFF
```

If the register save bit is not on, all index registers are preserved through the .EXIT. Thus, in the example above, when bit 35 is zero, there is no need to store an X4 value. Note that the instruction LREG must be avoided since it destroys X5, X6, and X7.

As in the .CALL and .GOTO routines, the A- and Q-register are stored in .STEMP+8 and .STEMP+9, respectively, during the .EXIT. If registers are not restored during the exit, then the routine which gets control must retrieve the A- and Q-registers from .STEMP+8 and .STEMP+9 before the inhibit bit is turned off.

CALLING SEQUENCE

The .EXIT macro may be called from either an HCM or an SSA module.

```
8           16
|-----------|-----------
| .EXIT     | n
```

where n is the number of instructions to be skipped when return is made following the calling sequence to the .CALL. A value of zero for n is the normal case; it is assumed when n is not given. This macro expands into:

```
8           16
|-----------|-----------
| XED       | .CREXT
| ZERO      | .SMDNO,1+n
```

The .EXIT routine obtains control through the pair of cells at .CREXT which contain the instructions:

	8	16
.CREXT	STC1	.SSTAK+2,5*
	TRA	HGT

OPERATING SYSTEM INTERACTION

The .EXIT routine marks, in the .CRMDD tables, the module that it leaves "idle." If the module is gated, another program may begin to execute the module. Under some circumstances, a program passing through .EXIT will be relinquished at the IC and I of the exit point. This is done in case there is a GEPR, SWAP/MOVE, ABORT, or courtesy call request in the state of the program (.STATE), and the program is returning to the main level of execution.

POSTCALLING SEQUENCE

When the .EXIT routine passes control to the returning module, the registers are restored if they were saved in .SREGS. Otherwise, they contain the values before the .EXIT routine was entered, except for the A- and Q-registers. The A- and Q-registers must be restored from .STEMP+8 and .STEMP+9.

SUPPORTING INFORMATION

The SSA may have been pushed down into the file in .SPDPA where the IC and I was made. If SSA is pushed down, the .EXIT routine pops the SSA back up before the return is made. When the SSA is read back, the IC and I stack and pointer which were pushed down are read back into core in the first five locations following the PATs. The stack is then transferred back to position with inhibited code after the I/O has completed.

Other Routines Used

Do Disc I/O Using System I/O Queue DMIOA (EP12 of .MDISP) issues the actual command when .EXIT must push down an SSA module.

Flowchart

See CPB-1500 for the flowchart of HEX, .MDISP module.

SYSTEM TRACE ROUTINE

The capability of tracing events occurring within the computer system is provided by the routine:

- TRACE System Trace

which is described in the following pages.

SYSTEM TRACE

TRACE (.MDISP), the System Trace routine, provides a record of system events in the form of a circular table. This circular table is useful in analyzing system dumps. Provision may also be made to dump this table on a tape file so that a record of system behavior may be made.

PRECALLING SEQUENCE

An entry may be made into the trace table at any convenient point. A trace entry is always two words in length. Bits 30-35 of the first word must be a code identifying the entry type. The rest of the data may be as desired. The format of the trace entries is described in GE-625/635 GECOS-III Introduction and System Tables, CPB-1488. The trace routine allows any set of trace entries to be turned on or off. Alternately, for maximum efficiency, all tracing may be turned off. Thus, when a trace entry is to be made, an in-line test must be made to determine if the trace is on or off. This test is as follows:

```
8        16
XED      .CRTRC
```

If the trace is off, control passes to the following location. This location ordinarily contains a transfer to the continuation of the user coding. If the trace is on, control passes to the second location beyond the XED. The coding for making the trace entry must be here. When control passes here, the following conditions exit:

1. The registers have been saved for restoration after the trace entry is made. Thus, registers (except index 0) may be modified when the trace entry is being constructed.
2. Only register X1 has been modified and must be used to enter the trace entry into the trace table.

An alternate method may be used to test if the trace is on or off. This method is:

```
8        16
XED      .CRTRC+1
```

In this case, registers are not saved, nor are any registers reloaded. This method is used when processing speed is of great importance. Control passes to the next location if the trace is off and skips one location if the trace is on.

In either case above, the user must construct the 2-word trace entry in A- and Q-registers. This may be done in any convenient manner. Many of the trace entries contain processor number and program number in bits 18-26 of the second word. Provision is made in the trace routine to add these fields to the Q-register, if desired. After the trace entry has been constructed in the A- and Q-registers, the trace entry is actually entered. The entry method depends upon the test made. For the case of the XED .CRTRC, the trace entry is made by TRA X,0. The value chosen for X determines if the processor and program number are to be put in the Q-register. If X = 0, they are; if X = 1, they are not. When the TRA X,0 control passes to the cell following the XED .CRTRC. When control passes to that location, all registers are restored to the value they had at the time of the XED .CRTRC.

If the XEC .CRTRC+2 test was used, X,0 is not set to return control to the trace entry routine. In order to make the trace entry, the following instructions are used:

```
8           16
LDX0        .CRTRC+3
TSX1        1,0*
```

If the processor and program number are to be put into bits 18-26 of the Q-register; the following is executed:

```
8           16
LDX0        .CRTRC+3
TSX1        0,0*
TSX1        1,0*
```

In contrast to the XED .CRTRC case, control passes to the instructions following the TSX1 1,0*. Registers are not restored.

CALLING SEQUENCE

TRACE is entered from any GECOS HCM or SSA module by one of two methods:

```
8           16
XED         .CRTRC
return
```

or:

```
8           16
TSX1        1,0*
return
```

OPERATING SYSTEM INTERACTION

The trace gate is used to sequence entries into the trace table.

ROUTINE RETURNS

After the trace entry is made, return is made to the location following the XED .CRTRC. At this time, all registers have been restored to the condition they held at the XED .CRTRC. If the trace was tested with a XEC .CRTRC, registers are not saved or restored. Registers X0, X1, X2, and the A- and Q-registers are destroyed when the trace is made.

SUPPORTING INFORMATION

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of TRACE .MDISP module.

SYSTEM GATE ROUTINES

The system gates are used to inhibit more than one processor executing certain sections of GECOS. This must be done to preserve the integrity of certain tables in the communication region or to execute non-reentrant procedures in the Hard Core Monitor. Two gate routines are used:

- OPGAT Open System Gate
- SHUG Shut System Gate

These routines are described on the following pages.

OPGAT
.MDISP

OPEN SYSTEM GATES

OPGAT (.MDISP), the .OPEN macro, opens a gate which has been previously shut with the .SHUT macro (see accompanying description, "Shut System Gates"). Opening a system gate allows a processor to execute a block of coding which may only be executed by one processor at a time. A gate is opened by the same processor which previously shut the gate.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

Three calling sequences are available to open system gates.

OPGAT is entered by an HCM or SSA module operating under a program number by:

```
8           16
.OPEN       GATE
```

which expands to the following:

```
8           16
NOP         GATE+1,SD
XED         .CROGT
```

OPGAT is entered by the Dispatcher (.MDISP) and the Interrupt Handler IOTRM (.MIOS), both of which operate under a program number of their own by:

```
8           16
.OPENS      GATE
```

which expands to the following:

```
8           16
EAX5        .CRGTC-.SGATE
NOP         GATE+1,SD
XED         .CROGT
```

The EAX5 instruction forces the system gate counters at .CRGTC to be used rather than those in .SGATE.

OPGAT is entered by an SSA module by:

```
8           16
.OPENP      GATE
```

The .OPENP call expands to the following:

```
8           16
------------------------
EAX0        GATE+1,$
STX0        1,IC
NOP         **,SD
XED         .CROGT
```

OPERATING SYSTEM INTERACTION

When a user shuts a gate, he inhibits any other processor from shutting the gate. If another processor were to attempt to do so, it will be delayed in a DIS instruction. A processor that has been held up because of an attempt to shut an already shut gate will be released to continue only when the other processor passes through the .OPEN routine.

When the contents of a table in the communication region are changed, that table generally must be protected from concurrent reference by another processor. Thus, the critical tables in the communication region are gated. A gate consists of a tally word of the form:

```
TALLYD  .CRGAT,0,2
```

This word is generally placed one location past the location symbol of the table. For instance, the gated table .CRJOB has the above tally word at address assembled to compensate for this.

Once a processor has shut a gate, it must not be interrupted until it has opened the gate. Thus, all coding executed while gates are shut must be inhibited.

ROUTINE RETURNS

User registers are not modified by .OPEN. Index register X0 is destroyed by .OPENP; index register X5 is destroyed by .OPENS. If a processor is waiting to shut a gate when another processor opens the gate, a CIOC is issued to all processors. The resulting connect fault in all processors is used to start the waiting processor. The connect fault vector in all processors has two NOP's in it. Thus, for a processor that is running, the fault is ignored. However, if the processor is in a DIS, that processor proceeds to the next instruction (XED) when the fault occurs. In this way, a processor that is waiting for a gate to open is started back up.

POSTCALLING SEQUENCE

None.

OPGAT
.MDISP

SUPPORTING INFORMATION

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of OPGAT, .MDISP module.

CLOSE SYSTEM GATES

SHUG (.MDISP), the .SHUT macro, shuts a gate which has been previously opened with the .OPEN macro (see accompanying description, "Open System Gates"). Closing a system gate allows only one processor to execute a block of coding which may only be executed by one processor at a time. A gate is shut by the same processor which previously opened the gate.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

Three calling sequences are available to close system gates.

SHUG is entered by an HCM or SSA module operating under a program number by:

```
8           16
┌──────────┬──────
| .SHUT    | GATE
```

which expands to the following:

```
8           16
┌──────────┬──────
| XED      | GATE+1,AD
| NOP      | GATE+1,SD
| XED      | .CRGAT+10
```

SHUG is entered by the Dispatcher (.MDISP) and the Interrupt Handler IOTRM (.MIOS), both of which operate under a program number of their own, by:

```
8           16
┌──────────┬──────
| .SHUTS   | GATE
```

which expands to the following:

```
8           16
┌──────────┬──────
| EAX5     | .CRGTC-.SGATE
| NOP      | GATE+1,SD
| XED      | .GROGT
```

The EAX5 instruction forces the system gate counters at .CRGTC to be used rather than those in .SGATE.

SHUG is entered by an SSA module by:

```
8           16
┌──────────┬──────
| .SHUTP   | GATE
```

The .SHUTP call expands to the following:

```
8           16
+-----------+-----------+
| EAX0      | GATE+1,$  |
| STX0      | 2,IC      |
| STX0      | 2,IC      |
| XED       | **,AD     |
| NOP       | **,SD     |
| XED       | .CRGAT+10 |
+-----------+-----------+
```

OPERATING SYSTEM INTERACTION

When user shuts a gate, he inhibits any other processor from shutting the same gate. If another processor were to attempt to do so, it will be delayed in a DIS instruction. A processor who has been held up because of an attempt to shut an already shut gate will be released to continue only when the other processor passes through the OPEN routine.

When the contents of a table in the communication region are changed, that table generally must be protected from concurrent reference by another processor. Thus, the critical tables in the communication region are gated. A gate consists of a tally word of the form.

```
TALLYD  .CRGAT,0,2
```

This word is generally placed one cell past the location symbol of the table. For instance, the gated table .CRJOB has the above tally word at .CRJOB+1. The various gating macros automatically add 1 to the address assembled to compensate for this.

Once a processor has shut a gate, it must not be interrupted until it has opened the gate. Thus, all coding executed while gates are shut must be inhibited.

ROUTINE RETURNS

User registers are not modified by .SHUT. Register X0 is destroyed by .SHUTP. Register X5 is destroyed by .SHUTS. If a processor is waiting to shut a gate when another processor opens the gate, a CIOC is issued to all processors. The resulting connect fault in all processors is used to start the waiting processor. The connect fault vector in all processors has two NOPS in it. Thus, for a processor that is running, the fault is ignored. However, if the processor is in a DIS, that processor will proceed to the next instruction, XED, when the fault occurs. In this way, a processor that is waiting for a gate to open is started back up.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Other Routines Used

None.

Flowchart

See CPB-1500 for the flowchart of SHUG, .MDISP module.

4. INTRODUCTION TO PERIPHERAL ALLOCATION

The Peripheral Allocator schedules and allocates all peripherals used by a slave program, an Input Media Conversion (IMCV) tape, a time-sharing program, and GEPR to exchange magnetic tape units. The following modules perform these functions:

- .MALC1 Peripheral Allocator
- .MALC2 Peripheral Dispenser
- .MALC5 Peripheral Space Allocator
- .MALC6 LLINK Allocator
- .MALC7 Permanent File Allocator

Also included are two MME processors used to request additional peripherals and release peripherals and/or memory:

- .MMORE MME GEMORE
- .MRELS MME GERELS

When a new job is received, the job control file (J*) is scanned to determine whether the job deck was made up correctly and which, if any, magnetic tape reels should be retrieved. During this scan, the number of executions and number of $ BREAK cards are accumulated for later decision making. The worst case is determined for memory requirements and magnetic tape needs, and process time and output line limits are accumulated. These values are then optionally used to set urgency and to gauge job length.

At the end of the initial scan of the job control file, the job is deleted if any errors were found, is put in limbo if special resources need retrieval, or is made a candidate for allocation if neither of the above is true. If the job is a lengthy one and is not an urgent job, it is bypassed (sieve status), and the operator informed. If a RUN request is not received within five minutes, the operator is reminded of the job status.

The first time an activity is considered for allocation, the J* file is scanned from its current position to the next activity delimiter to accumulate exact memory and peripheral requirements. The file control card information is extracted and condensed into a peripheral requirement summary by type of device. In addition, a peripheral detail entry is constructed for each file specified by a control card or implied because of the type of activity. The peripheral summary is used to perform a gross resource test before any detailed allocation is attempted; the detail entries are used to eliminate the need for performing a character scan of variable fields each time allocation is tried. Keeping a queue of jobs waiting for resources and attempting allocation more than once for each activity prevents hardware delays and maintains a high level of throughput.

Actual allocation is done by matching the details of peripherals needed to the resources available as described in the System Configuration Tables (SCT) until all needs are satisifed. During this process a Peripheral Assignment Table (PAT) is generated. The PAT allows .MIOS to associate a user-specified file code with an entry in the SCT and, at I/O time, a specific piece of peripheral hardware.

When allocation is successfully completed, an SSA image is written to the first 640 words of the job data file (*J), and an entry is made in the core allocation queue so the activity can be loaded and executed. At that time, the job status is set to "allocation complete" and is not considered again for allocation until the current activity has terminated.

In allocating files to devices that permit a choice of unit, such as magnetic tape and mass storage, the choices are based on channel of least usage for magnetic tape, and file unit of least usage for mass storage. Usage is defined as the number of files (regardless of size) currently allocated to the channel or unit being considered.

If a file serial number (FSN) is present on the $ TAPE control card, the scan is for a unit in Standby status on which the specified reel may be mounted. If a FSN is not present, the scan is for a unit in Ready status (scratch tape mounted and write-permit ring in) so that no operator action is necessary.

When a tape unit of proper status is not available, a second scan is made to pick a device which can be used. In this scan RDY/STBY is not considered, but the device must be on line with power on and be logically assignable but not already assigned as determined by the SCT. If a named device is designated at startup time and is specified as being dedicated, it is only allocated when requested by name.

5. ALLOCATION MODULES

PERIPHERAL ALLOCATOR (.MALC1)

The Peripheral Allocator (.MALC1) performs job scheduling. It scans the job control file (J*) and the job data file (*J) which have been prepared as described in the Introduction to Peripheral Allocation and in CPB-1490, GE-625/635 GECOS-III System Input.

The following routines comprise .MALC1:

- PANIC Fault Recovery
- ENTRY Initialize Program
- QUE Process Entries in Queue
- LOOP Scan Job Stack
- NEW Process New Jobs
- NUACT Determine Next Activity Requirement
- DOACT Make Default File Entries
- TALC Main Allocation Pass
- DONE Peripherals Allocated
- TAR Test for Tape Ready
- ALC2 Resident Peripheral Dispenser
- READJ Read J* File
- QIOS Fill Queues for Simple I/O
- TYPE Type Console Message
- FIND Scan Variable Control Field
- RELS Release Files in PAT

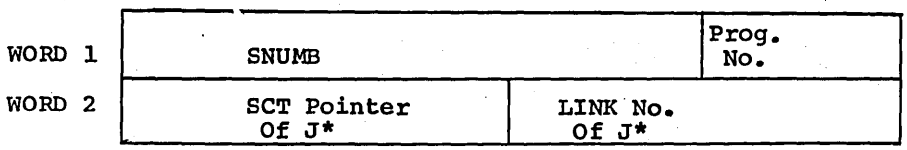
The SSA image is produced as the first two records (640 words) of the file. System configuration tables .CRCTi, .CRIOi, and various other communication cells, .CRxxx, are used to schedule and allocate jobs. The .CRSCT,.CRPOQ, and the .CRQGT gates are used.

Example

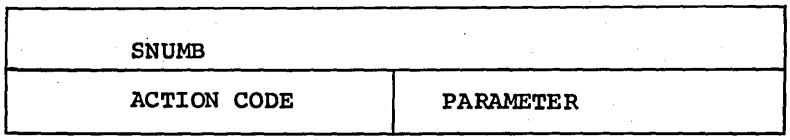
Requests are placed in the allocation input queue table and .MALC1 is enabled as shown below:

```
1       8           16
|       |           |
|       | INHIB     | ON
| ALPHA | .SHUT     | .CRJOB          Shut gate
|       | NOP       | .CRJOB+4,DI     Is there room in queue
|       | TTF       | BETA,$          Yes
|       | NOP       | .CRJOB          No, back up tally
|       | .OPEN     | .CRJOB          Open gate
|       | .GRELC    |                 Wait
|       | TRA       | ALPHA,$
| BETA  | LDAQ      | ARGUMENT,$      Get argument
|       | STAQ      | .CRJOB,AD       Make queue entry (see new job entry
|       |           |                 below)
|       | TTF       | 3,IC            Test tally runout
|       | LDA       | .CRJOB+3        Cycle tally
|       | STA       | .CRJOB
|       | .OPEN     | .CRJOB          Open gate
|       | INHIB     | OFF
|       | EAQ       | .PNALC          Program no.
|       | ORQ       | .CRPAU          Urgency value
|       | .CALL     | .MDISP,6        Enable .MALC1
|       | DRL       |                 Disaster return
|       | Continue  |                 Normal Return
```

The format of the 2-word input queue entry (pointed to by .CRJOB) is:



NEW JOB ENTRY



SUBSEQUENT ENTRIES

ACTION CODE	ACTION	PARAMETER
0	None	-
1	New Activity	Reason Code and Prog. No.
2	ENDJOB	Reason Code and Prog. No.
3	KILL/TERM	-
4	Hold Job	-
5	Run Job	-
6	New Urgency	Urgency
7	Not Used	

FAULT RECOVERY

PANIC (.MALC1) provides wrapup and restart of the Peripheral Allocator when a program fault occurs.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

PANIC is called from the Fault Processor if the fault is hardware. It is called from the Activity Termination Initiation module (.MBRT1) if it is a GECOS-detected fault.

Hardware Fault

```
8           16
TRA         .SICI,5
```

GECOS-Detected Fault

```
8           16
LDXO        23,5
TZE         2,IC
TSS         0,0
```

OPERATING SYSTEM INTERACTION

The top entry in the stack is the location of the fault/error.

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The exit is to location REENT in the Initialize Program routine, ENTRY (.MALC1).

POSTCALLING SEQUENCE

None.

PANIC
.MALC1

SUPPORTING INFORMATION

PANIC is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

PANIC occupies approximately 110 core storage locations.

Other Routines Used

Print Lines Prior to Snapshot BINPT (EP2 of .MSNP1)
Type Console Message TYPE (.MALC1)
Take Dump for System Abort FSB (EP1 of .MFALT)
Fill Queues for Simple I/O QIOS (.MALC1)
Process Entries in Queue QUE (.MALC1)

ENTRY
.MALC1

INITIALIZE PROGRAM

ENTRY (.MALC1) initializes PAT entries and counts to accumulate peripheral resources by type of device for gross allocation tests.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

Entry is called from the Dispatcher when a peripheral may possibly be allocated.

8	16
RET	.SICI,5

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The last instruction in ENTRY is a transfer to the Process Entries in Queue routine, QUE (.MALC1).

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

ENTRY is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

ENTRY occupies approximately 128 core storage locations.

Other Routines Used

None.

QUE
.MALC1

PROCESS ENTRIES IN QUEUE

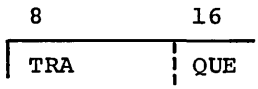
QUE(.MALC1) removes entries from the .CRJOB queue and takes appropriate action depending on type of entry found. It also maintains a job First-In First-Out (FIFO) status list for peripheral allocation scheduling.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

QUE is called from within the .MALC1 module when there are unprocessed entries in the .CRJOB queue.



OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRTSQ gate is used when a job is terminated from Time-Sharing.

ROUTINE RETURNS

The exit is to the Scan Job Stack LOOP (.MALC1) when all entries have been processed.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

QUE is nonreentrant.

Interrupts are inhibited while the .CRTSQ gate is shut.

Storage

No internal temporary storage is used.

QUE occupies approximately 350 core storage locations.

Other Routines Used

Type Console Message TYPE (.MALC1)
Write Execution Report Message EXCA (EP5 of .MYSOT)

LOOP
.MALC1

SCAN JOB STACK

LOOP (.MALC1) scans the FIFO list in order of urgency to determine which jobs to try to allocate. It is the primary job scheduler portion of GECOS-III.

PRECALLING SEQUENCE

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

X2 Index into FIFO list

CALLING SEQUENCE

LOOP is called from within the .MALC1 module when the queue is empty.

8	16
TNZ	LOOP

or:

8	16
TNC	LOOP

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The exit is to the Relinquish Control Until Program Enabled DSCNT (EP11 of .MDISP) when all jobs have been processed. A 2-minute alarm is set before giving up control.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

LOOP is nonreentrant.

Interrupts are inhibited when making entries for the Core Allocator and setting the alarm.

Storage

No internal temporary storage is used.

LOOP occupies approximately 220 core storage locations.

Other Routines Used

Type Console Message TYPE (.MALC1)
Enable Program ENB (EP6 of .MDISP)
Relinquish Control Until Program Enabled DSCNT (EP11 of .MDISP)
Set Alarm SCK (EP13 of .MDISP)

NEW
.MALC1

PROCESS NEW JOB

NEW (.MALC1) scans the job control file (J*) to check for format errors and job consistency, to determine worst tape case and presence of other peripherals required, to look for tapes to be retrieved from the tape library, and to examine maximum job core needs, total job time, and total job output for selective job scheduling.

PRECALLING SEQUENCE

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

X4 Pointer to 5-word job control entry

CALLING SEQUENCE

NEW is called from within the .MALC1 module when a new job is encountered.

8	16
TNZ	NEW

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The exit from NEW is to location NEXT in the Scan Job Stack routine, LOOP (.MALC1).

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

NEW is nonreentrant.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for maintaining comprehensive information regarding job parameters.

NEW occupies approximately 770 core storage locations.

Other Routines Used

Type Console Message TYPE (.MALC1)
Scan Variable Control Fields FIND (.MALC1)
BCD to Binary Conversion BCD (.MALC1)
READ J* File READJ (.MALC1)
Process End of Job EOJ (EP4 of .MYSOT)
Write Execution Report Message EXCA (EP5 of .MSYOT)
MME GEINOS Processor INOS (EP5 of .MIOS)

NUACT
.MALC1

DETERMINE NEXT ACTIVITY REQUIREMENT

NUACT (.MALC1) makes a one-time scan of the control file for the current activity of a job to determine memory and peripheral requirements for that activity. Variable field formats are scanned and converted into fixed field peripheral detail entries in the control stack record.

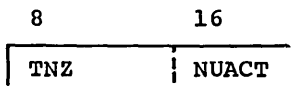
PRECALLING SEQUENCE

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

X4 Pointer to the 5-word job control entry

CALLING SEQUENCE

NUACT is called from within the LOOP routine of the .MALC1 module when an activity is examined for the first time.



OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRPOQ gate is used when a job is deleted.

ROUTINE RETURNS

The exit from NUACT is to the Make Default File Entries, DOACT (.MALC1) when an activity delimiter is found.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

NUACT is nonreentrant.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for the job control file data and for the control stack and SSA image.

NUCAT occupies approximately 1520 core storage locations.

Other Routines Used

Read J* File READJ (.MALC1)
Scan Variable Control Fields FIND (.MALC1)
BCD to Binary Conversion BCD (.MALC1)
Process End of Job EOJ (EP4 of .MYSOT)
Enable Program ENB (EP6 of .MDISP)

DOACT
.MALC1

MAKE DEFAULT FILE ENTRIES

DOACT (.MALC1) makes peripheral detail entries for implicit files; that is, files which are necessary for an activity but are not specified by file control cards. It also summarizes peripheral needs by type of device for gross allocation checks.

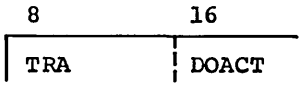
PRECALLING SEQUENCE

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

AR BCD mnemonic of next activity (activity delimiter)

CALLING SEQUENCE

DOACT is called from within the .MALC1 module after scanning the control cards for a given activity.



SUPPORTING INFORMATION

Programming Method

DOACT is nonreentrant.

Interrupts are not inhibited.

ROUTINE RETURNS

The exit from DOACT is to the Main Allocation Pass routine, TALC (.MALC1) if allocation is to be attempted. The exit is to the Scan Job Stack routine, LOOP (.MALC1) if allocation is damped.

SUPPORTING INFORMATION

Programming Method

DOACT is nonreentrant.

Interrupt are not inhibited.

Storage

Internal temporary storage is used for the job control file data and for the peripheral detail entries and SSA image.

DOACT occupies approximately 190 core storage locations.

Other Routines Used

None.

TALC
.MALC1

MAIN ALLOCATION PASS

TALC (.MALC1) performs detail allocation of peripherals to the specific file codes defined in the activity. It allocates new devices to new files and relates saved files to new file codes where possible.

PRECALLING SEQUENCE

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

X2 First-in, first-out (FIFO) index
X4 Pointer to 5-word job control entry

CALLING SEQUENCE

TALC is called from within the .MALC1 module when an activity is a candidate for allocation.

8	16	
TRA	TALC	If not new activity
TMI	TALC	If job is super critical

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The exit from TALC is to the Scan Job Stack routine, LOOP (.MALC1) when allocation has failed because of lack of resources. Exit is to the Peripheral Allocated routine, DONE (.MALC1) when allocation is successful.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

TALC is nonreentrant.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for a tentative PAT.

TALC occupies approximately 400 core storage locations.

Other Routines Used

Resident Peripheral Dispenser ALC2 (.MALC1)
Type Console Message TYPE (.MALC1)
Release Files in PAT RELS (.MALC1)

DONE
.MALC1

PERIPHERALS ALLOCATED

DONE (.MALC1) assigns pushdown space, does housekeeping regarding card reader, makes L* PAT entry, captures initial accounting data, prints or punches banners, and passes activity to the Core Allocator.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

DONE is called from the Main Allocation Pass routine TALC (.MALC1) when TALC has successfully completed peripheral allocation.

8	16
TRA	DONE

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRQCT gate is used when a card reader is allocated. The .CRPOQ gate is used when an entry is made for core allocation.

ROUTINE RETURNS

DONE exits to the Scan Job Stack routine, LOOP (.MALC1) when processing is complete.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DONE is nonreentrant.

Interrupts are inhibited when gates are shut.

Storage

Internal temporary storage is used for the slave PAT and SSA image and for initial accounting data.

DONE occupies approximately 520 core storage locations.

Other Routines Used

Resident Peripheral Dispenser ALC2 (.MALC1)
Type Console Message TYPE (.MALC1)
Fill Queues for Simple I/O QIOS (.MALC1)
Enable Program ENB (EP6 of .MDISP)
Write Execution Report Message EXCA (EP5 of .MYSOT)

TAR
.MALC1

TEST FOR TAPE READY

TAR(.MALC1) examines the PAT for tapes which required operator action (MNT/RDY) and determines if the requested action has been done. The Fill Queues for Simple I/O routine QIOS (.MALC1) is used to rewind tape units for status determination.

PRECALLING SEQUENCE

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

X2 Index into FIFO list
X4 Pointer to 5-word job control entry

CALLING SEQUENCE

TAR is called from within the Scan Job Stack routine, LOOP (.MALC1) when a job is examined which has operator action pending.

8	16
TNZ	TAR

OPERATING SYSTEM INTERACTION

No .STEMP storage is used

No gates are used.

ROUTINE RETURNS

The exit is to location CORE in the Peripherals Allocated routine, DONE (.MALC1) if all tapes are ready. The exit is to the Scan Job Stack routine, LOOP(.MALC1) if tapes are not ready.

SUPPORTING INFORMATION

Programming Method

TAR is nonreentrant.

Interrupts are not inhibited.

Storage

Internal temporary storage is used for preparation of operator reminder instructions when needed.

TAR occupies approximately 200 core storage locations.

Other Routines Used

BCD to Binary Conversion BCD (.MALC1)
Type Console Message TYPE (.MALC1)
Fill Queues for Simple I/O QIOS (.MALC1)

ALC2
.MALC1

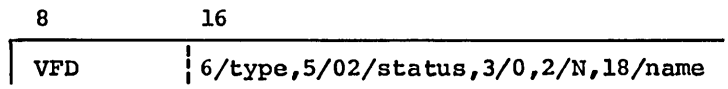
RESIDENT PERIPHERAL DISPENSER

ALC2(.MALC1) is a reasonable facimile of the GECOS module .MALC2. ALC2 is used very often and would require excessive pushdown and popups of the SSA if it were not included as a resident subroutine. ALC2 is identical in function to .MALC2 except that tape unit allocation is done cyclically rather than linearly.

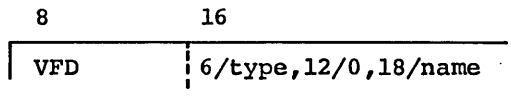
PRECALLING SEQUENCE

Prior to entering ENTRY, the A-register must contain a parameter word. The parameter word is of the form:

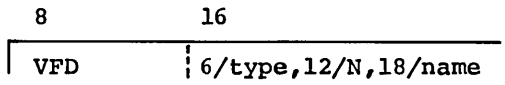
Magnetic Tape



Unit Record Devices



Mass Storage Links



where:

type is a 6-bit device type code in bits 0-5; values as shown below:

01	Disc Storage Subsystem (DSU200)
03	Magnetic Drum Subsystem (MDS200)
08	Magnetic Tape
09	7-Track ASA Magnetic Tape
10	9-Track ASA Magnetic Tape
16	Card Reader
17	Dual-Stacker Reader
18	CPZ100 Card Punch (100 cpm)
19	CPZ200 Card Punch (300 cpm)
20	CPZ201 Card Punch (300 cpm, flat-bed)
22	Printer
23	Paper Tape
24	Console

Bits 6-10 must be zero.

status is a 2-bit code in bits 11 and 12, applying only to magnetic tape:

0	Causes a search for a unit which is ready,
1	Looks for a unit in standby,
2 or 3	Implies either status is acceptable.

Bits 13-15 must be zero.

N bits (bits 16 and 17) is the number of devices requested. For mass storage allocation, n is the number of links desired. If the request is for magnetic tape and

n = 2 or 1	The request is for a single unit
n = 2 or 3	The request is for a dual unit.

Bits 18 through 35 contain the name, a 3-character logical unit designator, which has a non-numeric second character. In case a name is present, the type field is not necessary.

CALLING SEQUENCE

ALC2 is called from within the .MALC1 module either from the Main Allocation Pass routine, TALC, or from the Peripherals Allocated routine, DONE, when detail allocation of specific peripherals is wanted.

```
8          16
|----------|-----------------------------------
| LDA      | Argument
| STC2     | RETU,$          Set exit
| TRA      | ALC2,$          Enter routine
| return   |
|----------|
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRSCT gate is used when manipulating system tables.

ROUTINE RETURNS

The return is to the first instruction following the calling sequence. Registers contain:

X1 SCT pointer of primary device assigned, X1=0 not assigned

X0 SCT pointer of secondary device assigned, X0=0 not assigned

If request was for mass storage:

AR ZERO FIRST LINK, #LINKS

QR ZERO -1,-1 if single string sufficient or
ZERO FIRST LINK,#LINKS of second string

ALC2
.MALC1

POSTCALLING SEQUENCE

To test for allocation:

```
8                16
EAX1             0,1
TZE              denial routine
success routine
 .
 .
 .
TRA
```

SUPPORTING INFORMATION

Programming Method

ALC2 is nonreentrant

Interrupts are inhibited when the .CRSCT gate is shut.

Storage

Internal temporary storage is used for a preference list of devices to be examined for availability.

ALC2 occupies approximately 330 core storage locations.

Other Routines Used

None.

READ J* FILE

READJ(.MALC1) reads the current J* block of 320 words into the J* buffer area (BUFF) and checks for I/O errors.

PRECALLING SEQUENCE

Prior to entering READJ, the core storage cells listed must contain the data indicated.

RELB relative block number of the J* block to be read

JPOS SCT pointer and link number of the J* file

CALLING SEQUENCE

READSJ is called from within the .MALC1 module only when the allocator needs to read another block from J*.

```
8          16
┌──────────┬──────
│TSX1      │READJ
│End-of-file return
│Normal return
└──────────┘
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

An end-of-file return is to the first instruction following the TSX. The normal return is to the second instruction following the TSX. If an unrecoverable I/O error occurs, a DRL is executed.

POSTCALLING SEQUENCE

None.

READJ
.MALC1

SUPPORTING INFORMATION

Programming Method

READJ is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

READJ occupies approximately 32 core storage locations.

Other Routines Used

None.

FILL QUEUES FOR SIMPLE I/O

QIOS (.MALC1) gets an I/O queue using QUEUE (EP4 of .MIOS) and fills it in accordance with IOS rules so that I/O can be initiated by means of LINK (EP1 of .MIOS). GEPR override is used.

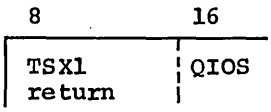
PRECALLING SEQUENCE

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

QR I/O command to be executed
X0 SCT Pointer of device

CALLING SEQUENCE

QIOS is called from within the .MALC1 module to perform simple I/O.



OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

If I/O has been done, the return is to first instruction following the TSX.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

QIOS is nonreentrant.

Interrupts are inhibited while filling the I/O queue entry.

GEPR override is used and all error statuses are the responsibility of the caller. The cells QSTAT, QSTAT+1 will contain such statuses upon return.

QIOS
.MALC1

Storage

No internal temporary storage is used.

QIOS occupies approximately 32 core storage locations.

Other Routines Used

Assign an I/O Entry QUEUE (EP4 of .MIOS)
Relinquish RLC (EP4 of .MDISP)
Link I/O to End of Queue LINK (EP1 of .MIOS)

TYPE CONSOLE MESSAGE

TYPE (.MALC1) types a specified message on one of the system consoles by using the Master Message Processor ITYM (EP7 of .MIOS).

PRECALLING SEQUENCE

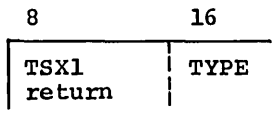
Prior to entering TYPE, the registers listed must contain the data indicated.

AR DCW of message to be typed (GITYM format)
X0 Message type for ITYM (EP7 of .MIOS)

The status return must be provided immediately in front of message.

CALLING SEQUENCE

TYPE is called from within the .MALC1 module only when the Allocator has a message for the operator.



OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The return is to the first instruction following the TSX when the message has been queued.

POSTCALLING SEQUENCE

TYPE does not wait for I/O to be completed. Therefore a roadblock should be used or the status return words tested for completion of the I/O where needed.

SUPPORTING INFORMATION

Programming Method

TYPE is nonreentrant.

Interrupts are not inhibited.

TYPE
.MALC1

Storage

No internal temporary storage is used.

TYPE occupies approximately 25 core storage locations.

Other Routines Used

Assign an I/O Entry QUEUE (EP4 of .MIOS)
Relinquish RLC (EP4 of .MDISP)
Master Message Processor ITYM (EP7 of .MIOS)

SCAN VARIABLE CONTROL FIELDS

FIND (.MALC1) scans the variable field (cols 16-72) of $ control cards to find either 1) a specified field, or 2) the next field.

PRECALLING SEQUENCE

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

X2 Index to beginning of card image to be scanned (Col 1)

CALLING SEQUENCE

FIND is called from within the .MALC1 module when a new control card is encountered with information in the variable field pertinent to Allocation.

To find a specific field:

```
8          16
|----------|-----------
|LDX2      | L(CARD)
|TSX1      | FIND
|ZERO      | Field Number
|null return
|normal return
```

To find the next field:

```
8          16
|----------|-----------
|TSX1      | LQAF
|null return
|normal return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The return is to the instruction following the calling sequence if the field is null or blank.

If the field is found (normal return), the return is to the second instruction following the calling sequence.

Content of the variable field is left-justified in IMAG, IMAG+1; the first 6 characters are also in the Q-register. The A-register (bits 30-35) contains delimiting character of field.

FIND
.MALC1

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

FIND is nonreentrant.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

FIND occupies approximately 135 core storage locations.

Other Routines Used

None.

RELEASE FILES IN PAT

RELS (.MALC1) releases part or all of the resources described in the SSA PAT image area when an activity cannot be completely allocated or a job is being deleted.

PRECALLING SEQUENCE

Prior to entering RELS, the core storage location listed must contain the data indicated.

RELSW = 0 if only new files are to be released

RELSW ≠ 0 if all files are to be released

CALLING SEQUENCE

RELS is called only from within the .MALC1 module when allocation has been denied or the job has been deleted.

```
8          16

TSX1       RELS
return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

The .CRSCT gate is used when testing and altering the system configuration tables.

ROUTINE RETURNS

The return is to the first instruction following the TSX when deallocation has been done.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RELS is nonreentrant.

Interrupts are inhibited when gates are shut.

RELS
.MALC1

Storage

No internal temporary storage is used.

RELS occupies approximately 160 core storage locations.

Other Routines Used

Return Links DA01 (EP2 of .MALC5) is used for release of mass storage space.

PERIPHERAL DISPENSER (.MALC2)

The Peripheral Dispenser module (.MALC2) is used by GECOS-III functions that require allocation of peripheral devices. It will, on option, select a specifically named device, select a device on a designated channel, or select a device on the channel of least use for a specified type of device. The channel or device status is tested for assignability, availability and, if magnetic tape, device status. If the specified conditions are met, the channel or device is marked assigned. To facilitate channel load balancing, a usage count is kept of the number of files allocated to multiple use channels such as magnetic tape or mass storage. Operator messages such as RDY/MNT and peripheral preparation such as REW/REWS are the responsibility of the user.

The routines listed below comprise the Peripheral Dispenser module:

- ENTRY (EP1) Process Specific Device Request
- ENT2 (EP2) Process Specific Channel Request
- ENT3 (EP3) Process Any Card Punch Request

ENTRY (EP1)
.MALC2

PROCESS SPECIFIC DEVICE REQUEST

ENTRY (EP1 of .MALC2) processes requests for a device specified by name or type.

PRECALLING SEQUENCE

Prior to entering ENTRY, the A-register must contain a parameter word of the form:

Magnetic Tape

8	16
VFD	6/type,5/0,2/status,3/0,2/N,18/name

Unit Record Devices

8	16
VFD	6/type,12/0,18/name

Mass Storage Links

8	16
VFD	6/type,12/N,18/name

where:

type is a 6-bit type code in bits 0-5; values are shown below:

01	Disc Storage Subsystem (DSU200)
03	Magnetic Drum Subsystem (MDS200)
08	Magnetic Tape
09	7-Track ASA Magnetic Tape
10	9-Track ASA Magnetic Tape
16	Card Reader
17	Dual-Stacker Reader
18	CPZ100 Card Punch (100 cpm)
19	CPZ200 Card Punch (300 cpm)
20	CPZ201 Card Punch (300 cpm, flat-bed)
22	Printer
23	Paper Tape
24	Console

Bits 6-10 must be zero.

status is a 2-bit code in bits 11 and 12, applying only to magnetic tape:

0	Causes a search for a unit which is ready,
1	Looks for a unit in standby,
2 or 3	implies either status is acceptable.

Bits 13-15 must be zero.

N (bits 16 and 17) is the number of devices requested. For mass storage allocation, n is the number of links desired. If the request is for magnetic tape and

n = 0 or 1	The request is for a single unit.
n = 2 or 3	The request is for a dual unit.

Bits 18-35 contain the name, a 3-character logical unit designator which has a non-numeric second character. In case a name is present, the type field is not necessary.

CALLING SEQUENCE

ENTRY is called from any module requiring device allocation.

```
8          16
|.CALL     |.MALC2,1
|return    |
```

OPERATING SYSTEM INTERACTION

At time of call, the contents of the A- and Q-registers are placed in .STEMP+8 and .STEMP+9.

The .CRSCT gate is used when manipulating the system configuration tables.

ROUTINE RETURNS

The return is to the first instruction following the .CALL.

Responses to requests for peripherals are given in index registers X0 and X1, and .STEMP+8 and .STEMP+9.

Index register X1 contains the SCT pointer to the primary allocated device if the request was successful. If the request was not successful, index register X1 contains zero. Index register X0 contains identical information on the alternate tape unit if one was requested.

If a magnetic tape request was denied, .STEMP+9 indicates whether the denial was because of device unavailability (zero) or a result of incorrect unit status (nonzero).

For successful mass storage requests, .STEMP+8 and .STEMP+9 contain the link strings assigned. If a single string was sufficient, .STEMP+9 contains -1.

ENTRY (EP1)
.MALC2

POSTCALLING SEQUENCE

The following instructions should be executed immediately upon return to prevent the .STEMP cells from being destroyed by an interrupt.

```
8               16
LDAQ            .STEMP+8,5
```

To test for successful allocation:

```
8               16
EAX1            0,1
T               denial routine
success         routine
   .
   .
   .
```

SUPPORTING INFORMATION

Programming Method

ENTRY is nonreentrant and written in floatable code.

Interrupts are inhibited when gates are shut.

Storage

Internal temporary storage is used for sorting of channels or units of like devices for load balancing.

ENTRY occupies approximately 390 core storage locations.

Other Routines Used

Peripheral Space Allocator (.MALC5) when requests for mass storage are received.

PROCESS SPECIFIC CHANNEL REQUEST

ENT2 (EP2 of .MALC2) processes requests for a device on a specific channel.

PRECALLING SEQUENCE

Prior to entering ENT2, the A-register must contain a parameter word as shown in ENTRY (EP1 of .MALC2) except that NAME is replaced by the SCT pointer.

CALLING SEQUENCE

ENT2 is called from any module requesting a specific channel.

```
8          16
┌──────────┬──────────┐
│.CALL     │.MALC2,2  │
│return    │          │
└──────────┴──────────┘
```

For further information on ENT2, refer to the description of ENTRY (EP1 of .MALC2).

ENT3 (EP3)
.MALC2

PROCESS ANY CARD PUNCH REQUEST

ENT3 (EP3 of .MALC2) processes requests for a card punch. The card punch can be either a CPZ100, CPZ200, or CPZ201.

PRECALLING SEQUENCE

Prior to entering ENT3, the A-register must contain a parameter word as shown in ENTRY (EP1 of .MALC2) except that name is replaced by the SCT pointer.

ENT3 is called from any module requesting a card punch.

8	16
.CALL return	.MALC2,3

For further information on ENT3, refer to the description of ENTRY (EP1 of .MALC2).

PERIPHERAL SPACE ALLOCATOR (.MALC5)

.MALC5 is a slave service area module with four entry points used to maintain a 64-word in-memory table. The in-memory table is initialized by .MPOPM and describes available links for all mass random access devices in the system configuration. Maintenance includes providing links when requested and returning unused links to the available pool. The four entry points are:

- CA01 (EP1) Provide Links for New File
- DA01 (EP1) Return Links
- CQ00 (EP3) Provide Additional Links
- CA00 (EP4) Provide Contiguous Links

Only a single copy of .MALC5 may be in execution at any one time. This ensures that two processors are not simultaneously looking at the same in-memory table.

When a request to obtain links is received, the following steps are taken:

1. A check is made to ensure that the links are available on the device requested. If not, an error return is made to the user.

2. A check is made to ensure that the number of links requested will not reduce the available space below threshold level. If it will, another check is made to ensure that the links are for an extension to an existing file. If not, an error return is made to the user.

3. A search is made of the in-memory available link table for an entry equal to the requested number of links. If found, it is deleted from the table and returned to the user. This is the most desirable case since it reduces the active size of the in-memory table.

4. A search is made of the in-memory table for the first entry (by position from the top of the table) describing more than the requested number of links. If found, the requested links are deleted from this entry. This is the next most desirable case since it still gives contiguous allocation to the user. If the requests in steps 3 or 4 are not satisfied and contiguous allocation is requested, an error return is made to the user.

5. A search is made of the in-memory table for its largest entry (which by this step is still less than the requested number of links). It is set aside and deleted from the table and steps 3 and 4 are repeated for a reduced number of links. If 3 and 4 fail, the set aside entry is reentered into the table by simulating a call to return links to the available pool and then making an error return to the user.

When returning links to the available space pool, the following steps are taken:

1. The in-memory table is searched to see if the returned links concatenate with any present entry. If they do, they are properly attached and the adjacent entry is tested for possible concatenation. If it also fits, table size is reduced by one.

2. If the table has at least one available location, the table expands by one cell and the returned links are entered into the proper place in the table to maintain order by ascending link number.

3. If the table is full, the first entry describing the smallest number of links is eliminated from the table and the returned entry (unless it is smallest) is entered into the appropriate position. In this mode, space is actually eliminated from the available pool. This space is reclaimed at startup or reboot time since it is still accurately described on the device from which the in-memory table is initialized.

If a catastrophe occurs, the in-memory table is reinitialized; this effectively reclaims any lost temporary storage.

.MALC5 requires approximately 290 core storage locations.

PROVIDE LINKS FOR NEW FILE

CA01 (EP1 of .MALC5) allocates links to build a new file.

PRECALLING SEQUENCE

Prior to entering CA01, the registers listed must contain the data indicated.

AU	Pointer to SCT Table
QL	Number of links requested

For example:

```
8          16
┌──────────┬──────────────────
│LDA       │PAT Table Word 1
│LDQ       │N,DL
```

The upper half of the PAT table word contains the absolute address of the secondary System Configuration Table.

N is equal to the number of links desired.

CALLING SEQUENCE

CA01 can be called from any GECOS-III module.

```
8          16
┌──────────┬──────────
│.CALL     │.MALC5,1
│Error return
│Normal return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

If the request for space was satisfied by two link strings, their description appears in the A- and Q-registers. In addition, bits 0 and 2 in the A-register are on indicating that the file is in the rewound position and that the A-register does not contain the last descriptor of the file. The link strings in the A- and Q-registers are ready for immediate storage in the PAT. The return is to .CALL+2.

When the request cannot be filled, a return is made to .CALL+1. The data in the A- and Q-registers is meaningless.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

CA01 is nonreentrant and written in floatable code.

Interrupts are inhibited while checksumming the link table.

Storage

No internal temporary storage is used.

Other Routines Used

Take Dump for System Abort FSB (EP1 of .MFALT) if a checksum error occurs in the link table.

RETURN LINKS

DA01 (EP2 of .MALC5) releases and returns unused links to an available pool.

PRECALLING SEQUENCE

Prior to entering DA01, the registers listed must contain the data indicated:

AU	Pointer to SCT Table
QU	Address of start of STRING
QL	Count of elements in STRING

where STRING is of the form: Link Number, Number of Links

For example:

```
1               8         16
|               |         |
|               | LDA     | PAT Table word
|               | LDQ     | PARAM
|               |  .      |
|               |  .      |
|               |  .      |
| PARAM         | ZERO    | LST,N
|               |  .      |
|               |  .      |
|               |  .      |
| LST           | ZERO    | LINK#,#LINKS
|               |  .      |
|               |  .      |
|               |  .      |
| (LST+N-1)     | ZERO    | LINK#,#LINKS
```

The upper half of the PAT table word contains the absolute address of the secondary System Configuration Table.

The link strings in table LST may be PAT table words, therefore, bits 0, 1, and 2 are used. Bit 1 is checked to ensure that this description is not part of another file. If Bit 1 is on, the return to the available pool is ignored.

CALLING SEQUENCE

DA01 can be called from any GECOS module.

```
        8         16
        |         |
        | .CALL   | .MALC5,2
        | return  |
```

DA01 (EP2)
.MALC5

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

The return is to the first instruction following the .CALL.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

DA01 is nonreentrant and written in floatable code.

Interrupts are inhibited while checksumming the link table.

Storage

No internal temporary storage is used.

Other Routines Used

Take Dump for System Abort FSB (EP1 of .MFALT) if a checksum error occurs in the link table.

PROVIDE ADDITIONAL LINKS

CQ00 (EP3 of .MALC5) allocates links to extend an existing file.

PRECALLING SEQUENCE

Prior to entering CQ00, the registers listed must contain the data indicated.

AU	Pointer to SCT table
QL	Number of links requested

CALLING SEQUENCE

CQ00 is called from the .MMORE module.

```
        8       16
┌──────────┬──────────┐
│  .CALL   │ .MALC5,3 │
│ Error return        │
│ Normal return       │
└─────────────────────┘
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

If the request for space was satisfied by two link strings, they appear in the A- and Q-registers. In addition, bits 0 and 2 in the A-register are on indicating that the file is in rewound position and that the A-register does not contain the last descriptor of the file. The link strings in the A- and Q-registers are ready for immediate use in the PAT table. The return is to .CALL+2.

When the request cannot be filled, a return is made to .CALL+1. The data in the A- and Q-registers is meaningless.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

CA01 is nonreentrant and written in floatable code.

Interrupts are inhibited while checksumming the link table.

Storage

No internal temporary storage is used.

Other Routines Used

Take Dump for System Abort FSB (EP1 of .MFALT) if a checksum error occurs in the link table.

PROVIDE CONTIGUOUS LINKS

CA00 (EP4 of .MALC5) allocates contiguous links for a new file.

PRECALLING SEQUENCE

Prior to entering CA00, the registers listed must contain the data indicated.

AU	Pointer to SCT Table
QL	Number of links requested

CALLING SEQUENCE

CA00 is called from the .MFS10 module.

```
8       16
.CALL   .MALC5,4
Error   return
Normal  return
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

A link string in the form LINK number, number of LINKS is returned in the A-register with bit 2 on indicating that the file is in rewound position. This bit is turned on so that the link string is prepared for immediate use in the PAT table. The Q-register contains all bits. For an error return, the data in the A- and Q-registers is meaningless. An error return can be caused three ways:

1. Not enough available space on the device
2. Not enough available contiguous space on the device
3. Not enough available space for the creation of a new file (space on the file is reserved for the extension of existing files)

POSTCALLING SEQUENCE

None.

CA00 (EP4)
.MALC5

SUPPORTING INFORMATION

Programming Method

CA00 is nonreentrant and written in floatable code.

Interrupts are inhibited when checksumming the link table.

Storage

No internal temporary storage is used.

Other Routines Used

Take Dump for System Abort FSB (EP1 of .MFALT) if a checksum error occurs in the link table.

LLINK ALLOCATOR (.MALC6)

.MALC6 is an SSA module with one entry point. It is used to allocate LLINKs (320-word links).

When a request to obtain LLINK space is received, the bit table which describes the LLINK is read into core, checksummed, and searched for an available LLINK. If one is found, the absolute block number of the LLINK is calculated and returned to the calling program. The bit table is altered, rechecksummed, and written to the appropriate device.

If no LLINK space is available, the block number is set to zero and control is returned to the calling program.

Only a single copy of .MALC6 may be in execution at any one time. This ensures that two processors are not simultaneously trying to allocate LLINK space.

START (EP1)
.MALC6

ALLOCATE LLINK SPACE

START (EP1 of .MALC6) reads the available LLINK table from the appropriate device and allocates the next available LLINK.

PRECALLING SEQUENCE

Prior to entering START, the SCT of the appropriate device should be placed in .SGCPA,5.

CALLING SEQUENCE

START is called from .MFS01, MFS02, and .MFS07 modules.

8	16
LDXO	SCT address
STXO	.SGCPA,5
.CALL	.MALC6,1
LDQ	.STEMP+9,5

OPERATING SYSTEM INTERACTION

.STEMP+9 is used.

No gates are used.

ROUTINE RETURNS

Return is to the calling program.

.STEMP+9 will contain the absolute block number of the allocated LLINK if space is available; otherwise, it will contain all zeros to indicate that space on this device is not available.

POSTCALLING SEQUENCE

Upon return .STEMP+9 should be examined immediately to prevent destruction by an interrupt.

SUPPORTING INFORMATION

Programming Method

START is nonreentrant and written in floatable code.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

START occupies approximately 378 core storage locations.

Other Routines Used

Link I/O to End of Queue LINK (EP1 of .MIOS)
Assign an I/O Entry QUEUE (EP4 of .MIOS)
Relinquish Control RLC (EP4 of .MDISP)

PERMANENT SPACE ALLOCATOR (.MALC7)

.MALC7 is an SSA module with two entry points:

- FSUNDT (EP1) Make Link Space Permanent Given PAT
- FSUNLS (EP2) Make Link Space Permanent Given Link

When returning a LINK to the available space pool, the bit table which describes LINK space is read into core, checksummed, and modified to reflect the available block. It is then rechecksummed and written on the appropriate device.

MAKE LINK SPACE PERMANENT GIVEN PAT

FSUNPT (EP1 of .MALC7) makes allocated space permanent by taking space description out of the available link table. The contents of the A-register point to start of a PAT entry.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

FSUNPT is called from the .MFS04, .MFS05, and .MFS10 modules.

```
        8          16
┌──────────┬──────────────────────
│ EAA      ¦ Absolute PAT address
│ .CALL    ¦ .MALC7,1
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to the calling program.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

FSUNPT is nonreentrant and written in floatable code.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

FSUNPT occupies approximately 368 core storage locations.

Other Routines Used

Link I/O to End of Queue LINK (EP1 of .MIOS)
Assign an I/O entry QUEUE (EP4 of .MIOS)
Relinquish Control RLC (EP4 of .MDISP)
Obtain Normal Snapshot GESNIP (EP1 of .MSNP1)

MAKE LINK SPACE PERMANENT GIVEN LINK STRINGS

FSUNLS (EP2 of .MALC7) makes allocated space permanent by taking space description out of the available link table. The A-register points to the start of a link string.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

FSUNLS is called from the .MFS04, .MFS05, and .MFS10 modules.

```
    8          16
┌──────────┬──────────────────────────────────
│ EAQ      │ SCT pointer
│ EAA      │ Absolute address of link string
│ .CALL    │ .MALC7,2
```

OPERATING SYSTEM INTERACTION

No .STEMP storage is used.

No gates are used.

ROUTINE RETURNS

Return is to the calling program.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

FSUNLS is nonreentrant and written in floatable code.

Interrupts are not inhibited.

Storage

No internal temporary storage is used.

FSUNLS occupies approximately 368 core storage locations.

Other Routines Used

Link I/O to End of Queue LINK (EP2 of .MIOS)
Assign an I/O Entry QUEUE (EP4 of .MIOS)
Relinquish Control RLC (EP4 of .MDISP)
Obtain Normal Snapshot GESNIP (EP1 of .MSNP1)

MME PROCESSORS FOR ALLOCATION

During execution of an activity the user may wish to release or alter the disposition of a peripheral file, increase the size of a mass storage file, or obtain an additional magnetic tape or mass storage file. The release of files is effected by a MME GERELS; the allocation of additional space or files is done by a MME GEMORE. The actual work of processing these requests is done by two SSA modules, .MRELS and .MMORE, which are called by the .MFALT module when these MME's are executed.

MORE
.MMORE

MME GEMORE PROCESSOR

MORE (.MMORE) is an SSA module with one entry point. It services a MME GEMORE request for additional mass storage space or for magnetic tape handlers. Requests for more core storage are relayed to GEPOP.

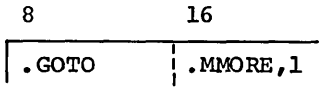
PRECALLING SEQUENCE

Bits 24-35 of the Q-register must contain the file code.

If request is for magnetic tape, bits 0-17 of the Q-register must either be zeros or a pointer to the core storage location that contains the file serial number (BCI) of the reel to be mounted.

CALLING SEQUENCE

MORE is called from the Fault Processor in response to a slave MME GEMORE.



OPERATING SYSTEM INTERACTION

The top entry in the control stack (.SSA) must contain the address of the MME GEMORE.

No .STEMP storage is used.

The .CRPOQ gate is shut while a core storage request is being relayed.

ROUTINE RETURNS

Return is made using the top entry in the stack by a .EXIT 2 if a peripheral request is satisfied and a .EXIT 1 if it is denied. If the .EXIT 2 is taken, the Peripheral Assignment Table contains the requested resource description.

If a memory request is received, MORE executes a .GOTO .MDISP,1 after relaying the request to GEPOP.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

MORE is nonreentrant but multiple copies may exist in core simultaneously. It is written in floatable code.

Interrupts are inhibited when the .CRPOQ gate is shut.

Storage

Internal temporary storage is used for preparation of operator instructions.

MORE occupies approximately 375 core storage locations in the lower half of the caller's SSA.

Other Routines Used

Process Specific Device Request ENTRY (EP1 of .MALC2)
Provide Additional Links CQ00 (EP3 of .MALC5)
Terminate Error Entry FALT (EP3 of .MBRT1)
MME GEINOS Processor INOS (EP5 of .MIOS)
Master Message Processor ITYM (EP7 of .MIOS)
Enable Program ENB (EP6 of .MDISP)
Relinquish Control Until Program Enabled DSCNT (EP11 of .MDISP)
Set Alarm SCK (EP13 of .MDISP)

RELS
.MRELS

MME GERELS PROCESSOR

RELS (.MRELS) is an SSA module with one entry point. It services a MME GERELS request for peripheral release or alteration of peripheral disposition.

PRECALLING SEQUENCE

None.

CALLING SEQUENCE

RELS is called from the Fault Processor module in response to a slave MME GERELS.

8	16
.GOTO	.MRELS,1

OPERATING SYSTEM INTERACTION

The top entry in the control stack (.SSA) must contain the slave address of the MME GERELS.

No .STEMP storage is used.

The .CRQGT gate is shut while resetting the GESPEC status of a released card reader.

The .CRSCT gate is shut while altering the configuration table.

ROUTINE RETURNS

Return is made to the top entry in the stack by a .EXIT after the stack entry has been adjusted by the length of the calling sequence.

POSTCALLING SEQUENCE

None.

SUPPORTING INFORMATION

Programming Method

RELS is nonreentrant but multiple copies may be in core simultaneously. It is written in floatable code.

Interrupts are inhibited when the .CRQGT and .CRSCT gates are shut.

Storage

No internal temporary storage is used.

RELS occupies approximately 450 core storage locations in the lower half of the caller's SSA.

Other Routines Used

MME GEINOS Processor INOS (EP5 of .MIOS)
Master Message Processor ITYM (EP7 of MIOS)
Return Links DA01 (EP2 of .MALC5)

INDEX

DOCUMENT REVIEW SHEET

TITLE: GE-625/635 GECOS-III Dispatcher and Peripheral Allocation

CPB #: 1491

From:

Name: _____

Position: _____

Address: _____

_____

Comments concerning this publication are solicited for use in improving future editions. Please provide any recommended additions, deletions, corrections, or other information you deem necessary for improving this manual. The following space is provided for your comments.

COMMENTS: _____

Please cut along this line

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
Fold on two lines shown on reverse
side, staple, and mail.

STAPLE

STAPLE

FOLD

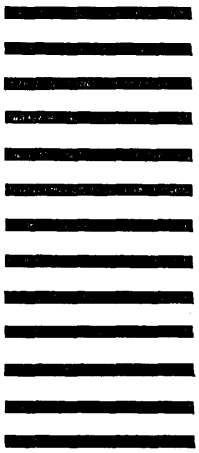
FIRST CLASS
PERMIT, No. 4332
PHOENIX, ARIZONA

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

GENERAL ELECTRIC COMPANY
PROCESSOR EQUIPMENT DEPARTMENT
13430 NORTH BLACK CANYON HIGHWAY
PHOENIX, ARIZONA 85029

ATTENTION: Program Documentation C-78
Systems and Processors Operation



FOLD

INFORMATION SYSTEMS

GENERAL ELECTRIC

LITHO U.S.A.