LEVEL 66/SERIES 6000

GENERAL COMPREHENSIVE OPERATING SUPERVISOR (GCOS)

SUBJECT

Revision to the *General Comprehensive Operating Supervisor (GCOS)* Manual.

SPECIAL INSTRUCTIONS

This is the first revision to DD19, dated April 1974. Change bars in the margins indicate technical changes and additions; asterisks in the margins indicate deletions.

SOFTWARE SUPPORTED

Series 60 Level 66 Software Release 4S3 and DPS1.3
Series 6000 Software Release JS3

ORDER NUMBER

DD19-01

August 1980

Honeywell

PREFACE

This manual describes the General Comprehensive Operating Supervisor (GCOS) software for the Honeywell Series 60 Level 66, Series 6000 and the Distributed Processing System (DPS 8) Information Processing System. Included in this manual are descriptions of GCOS software functions, a definition of the Program Switch Word, descriptions of the System Files, and a description of the Slave Program Prefix area.

In this manual, general terms are used to refer to the computer system when the text applies to both Series 60 Level 66, Series 6000 and DPS 8 systems. Where the text applies to one system or the other that system is identified in the text. Throughout this manual, the Series 60 Level 66 system is referred to as the Series 60 system.

File No.: 1P13,1713

DD19-01

LISTING OF MANUALS

A listing of large system software manuals is available to any Honeywell user who has access to an upper and lower-case ASCII terminal with a line length of 80 or more characters. The manuals are categorized both by software release and by software category. This listing is updated regularly to enable ordering of manuals as soon as they are published. Instructions on how to order manuals are output with the listing.

To obtain the listing:

1. Dial appropriate telephone number to connect your terminal with the Multics system in Phoenix.

300-baud terminals		150-baud terminals
(602)249-7556	249-7701	249-7554
249-7501	249-7614	

   System response - computer system identification

```
Example:  Multics MR6.0:  Honeywell LISD Phoenix, System M
          Load=35.0 out of 125.0 units:  users=35
```

2. Enter the following login command: $\boxed{\text{login Sam}}$ (the identifier "Sam" must be used - it is not a sample for any proper name)

   Press carriage return key.

   System response - request for password

```
Example:  Password:
          ██████████████████████ (password strikeover mask)
```

3. Enter the password: $\boxed{\text{Multics}}$

   Press carriage return key.

   System response - informative message of the day and "r" message

```
Example:  The system will shut down Washington's birthday.
          r1111.7 Thu (ready message)
```

4. To obtain a list of software manuals, enter: $\boxed{\text{:manrep}}$

   System response - report is printed at terminal

   When END OF REPORT is reached, the list is complete.

5. To log off Multics system, enter: $\boxed{\text{logout}}$

   System response - logout message

```
Example:  Sam SRB logged out 07/04/80 1110.3 mst fri
          CPU usage 3 sec, memory usage 22.4 units
```

CONTENTS

CONTENTS (cont)

CONTENTS (cont)

CONTENTS (cont)

CONTENTS (cont)

ILLUSTRATIONS

CONTENTS (cont)

SECTION I

INTRODUCTION

Operating under control of the General Comprehensive Operating Supervisor (GCOS) software, a multi-dimensional Honeywell Series 60 Level 66, Series 6000, or DPS 8 Information Processing System[1] provides the user with a full range of business and scientific computational capability. The versatility of GCOS allows it to perform the following functions:

- Integrate local batch processing, remote batch, remote access, transaction processing, interactive remote job entry, online document handling, and time-sharing for maximum system utilization.

- Provide a common file system with file protection and access control, which is accessible from all modes of processing.

- Increase the total throughput of the computer (the amount of work that may be performed in any given time).

- Reduce user program turn around time in large-scale installations (elapsed time from program submission to the machine room up to return of program solutions).

- Provide logical, easy-to-use programmer and operator interfaces with system software.

- Maximize system availability through a comprehensive Total Online Testing System.

To perform these functions with the efficiency required for multi-dimensional operation, GCOS software serves as the overall manager for the operation of the integrated hardware/software system. As system manager, it achieves maximum utilization of the total hardware system and supervises the multiprocessing/multiprogramming environment, which is the normal operating mode of the system. Significant GCOS software features required to perform these functions are implemented as modules, programs and subroutines within GCOS software.

[1] In this manual, the specific system is named when the text applies only to that system.

These GCOS software features include:

Maximum system throughput via multiprogramming
Multiprocessor control
Total Online Testing System
Concurrent time-sharing operation
File Management Supervisor
* Cache Memory
Multiple local and/or remote job input streams
Scheduling and coordination of jobs
Input/output supervision on an interrupt-oriented basis
File-oriented programming (instead of device oriented)
Fault detection with standard GCOS software or optional programmer-supplied corrective actions
Maximum efficiency of main memory use through dynamic program relocation and system-controlled subprogram overlays
Modular construction for simplified maintenance

GCOS SOFTWARE OPERATIONAL ENVIRONMENT

GCOS software executes on Series 6000 (EIS only) and 6000N hardware. GCOS also supports Series 60 Level 66 Systems and the Distributed Processing System (DPS 8).

CPL/NPL Peripheral Device Coexistence

CPL and NPL peripheral devices may coexist within Series 6000N and Series 60 Level 66 and Level 66/DPS (with BCD option) systems in either single system or shared mass store configurations. However, only CPL devices may exist within Series 6000 systems.

Multiprogramming

Although each user programmer writes a job program as though it had exclusive use of the computer, the generated program resides concurrently in main memory with other user programs and is executed in a time-shared manner. That is, any given program is processed until it is held up (usually because of the need for some input/output function to be completed). The next most urgent program is then processed. The ways in which a user program can be temporarily delayed in execution are:

- o Roadblock - Program cannot progress until all input/output requests have terminated.
- o Relinquish - Program relinquishes control until any one of its input/output requests has terminated.
- o Forced Relinquish - Program was interrupted because a timer runout or I/O interrupt occurred.
- o MME GEWAKE - Program relinquishes control until a specified time has elapsed.

Each time a program yields control by means of Roadblock, Relinquish, Forced Relinquish, or MME GEWAKE, GCOS gives control to another program, which can make effective use of the processor.

In giving such control, GCOS examines the following conditions:

- o Program urgency (priority) compared to other programs that reside in memory
- o Roadblock status involving completion of all input/output
- o Completion of input/output that was pending when the last Relinquish was given
- o Request present for use of the processor.

On Line Media Conversion

Media conversions are of two basic types; (1) bulk media conversion, whereby large volumes of data in a single format and for a single purpose are processed, and (2) system media conversion, where low-volume sets of data, each with its own format and purpose, are processed.

Bulk media conversion is performed by the Bulk Media Conversion (BMC) system program, which is called into execution by a control statement. Other control statements provide BMC with the location (media) of the data to be converted and designate the device to be used for data output. Use of BMC is described in the Bulk Media Conversion reference manual.

On line system media conversions for both input and output are performed as a normal part of job processing. Normal job input is performed by input media conversion, which reads job input from card readers, magnetic tape, file system, mass storage, and remote devices concurrently; scans the control statements for execution information; and records the jobs on the input queue located on system storage.

System media conversions of program output data are automatically performed by the output media conversion routines, which execute in protected memory. The programmer specifies that a particular output file be written on the permanently assigned system output (SYSOUT) file. He uses the print, punch, or write record functions of the File and Record Control facility to do this. Once on the SYSOUT file, the output is converted to hard copy or punched cards by output media conversion. This operation is performed concurrently with output of other user programs, which are executing in the multiprogramming environment.

Centralized Input/Output

In the multiprogramming environment, several programs may concurrently request input/output. These multiple requests must be processed in a manner that provides efficient use of the entire complement of peripheral devices and at the same time maintains the continuous processing of other programs in the system. The Input/Output Supervisor program (IOS) of GCOS performs this function.

The main functions of IOS are to initiate an input/output activity and to respond to the termination of that activity. In addition, IOS provides the following functions:

- o File code to physical unit translation
- o Protection of user files
- o Simulated tape processing on disk
- o Supervision of all input/output interrupts
- o Queueing of input/output requests
- o Utilization of crossbarred peripheral channels
- o Maintaining of an awareness of the status of each peripheral device.
- o Accounting of time spent by the processor and all peripheral devices for each program executed.

When IOS receives a request to perform an input/output function, it determines whether a particular channel is busy. If the channel is not busy, IOS issues a Connect instruction. If the particular channel is busy, the request is placed in a waiting queue. If the request queue for that program is full or if the program indicated should be roadblocked until all input/output is complete, control is given to another program residing in memory.

When the input/output operation terminates, control is given to IOS to perform all necessary termination functions. At this point, the request queue is examined. If any requests for the channel are in queue, they are performed. If applicable, latency reduction techniques are applied to maximize mass storage effectiveness.

Master/Slave Relationship

The processor unit of the computer system operates in either master or slave mode. GCOS software executes in both modes, as required. When executing a user program, a processor is in slave mode. When executing GCOS software, it operates in master mode.

The primary reason for master mode operation is to protect GCOS software and user programs from modification by other user programs. This feature is vital in the multiprogramming environment and is closely tied in with memory protection, accounting determinations, multiprogram interrupt management, intermodule communications control, and input/output operations. Each of these functions is implemented by a processor instruction that executes in master mode.

All instructions available to the processor in slave mode are available in master mode, but master mode instructions are not available to slave mode programs. The following instructions can be executed only in master mode.

- o Load Base Address Register (LBAR)
- o Load Timer Register (Load T)
- o Load Base Extension Register (Load BER)
- o Load Master BAR A (Load MBA)

- Load Master BAR B (Load MBB)
- Load Central Processor Registers (Load CPR)
- Set Memory Controller Interrupt Cells (SMIC)
- Store Central Processor Registers (SCPR)
- Read Memory Controller Mask Registers (RMCM)
- Read System Controller Registers (RSCR)
- Set Memory Controller Mask Registers (SMCM)
- Set System Controller Registers (SSCR)
- Delay Until Interrupt Signal (DIS)
- Connect Input/Output Channel (CIOC)

NOTE: The MLDA, MLDQ, MLDAQ, MSTA, MSTQ, and MSTAQ, for systems with memory capacity greater than 256K words, are modified LDA, LDQ, LDAQ, STA, STQ, and STAQ instructions and can only be executed in master mode.

The Connect Input/Output Channel instruction is the beginning of every peripheral operation. Thus, all peripheral operations are performed in master mode by the Input/Output Supervisor within GCOS software.

Master mode operation by the processor is a primary safeguard for executive routines and user programs in memory. The applications programmer can force the processor into this mode, but only to access routines that are part of GCOS software. This is through the use of the Master Mode Entry (MME) instruction and one of the system-symbol operands described in Section IV. (Any other use of the MME instruction causes an abort of the user program.) Thus, through the MME instruction, the programmer can communicate with GCOS software modules to exchange any information necessary for the execution of his program.

Hexadecimal Floating Point

The DPS 8 Processors have hexadecimal floating point capability for slave programs. The hexadecimal mode is disabled if any processors in the system do not have hex mode capability. An assign of a processor that does not have hex mode capability is not allowed on a system that has been initialized with the hex mode option. During system initialization all processors are interrogated to determine if they have hex mode capability. If they do, bit 28 of .CRFIG is set to a one state. Hex mode is enabled by MME GMODES and is disabled by MME GMODER.

Mass Storage Orientation

Computer overhead time is reduced and multiprogramming is enhanced through the use of mass storage subsystems for system storage.

Shared Mass Storage

The shared mass storage facility can enhance system operations by permitting two systems to share mass storage resources. Two independent computer systems, each operating under its own GCOS software, can share a common data base on mass storage devices configured for the mass storage environment. In addition, a single system or a pair of sharing systems can be configured to permit GCOS software and NPS software to share space on a designated mass storage device.

In the shared mass storage environment, the two systems share a common System Scheduler file thereby permitting load levelling between the systems. Also in this environment, the System Output program (SYSOUT) provides the capability for moving system output from one system to the other, either through console instructions or by requests embedded in the user's program.

To ensure file security with two sets of users accessing the same data base, the shared mass storage facility includes a complex set of access controls to protect files and to prevent unauthorized access to files.

The shared mass storage facility also enhances system reliability. The impact of an abort in one system is minimized by permitting the other system to continue operating and in specified cases to assume output responsibility for the aborted system.

*

Program File Orientation

The software system is further described as file oriented because GCOS assigns peripheral devices to an activity by file code designators and manages assigned peripherals during input or output operations. Thus the programmer never deals directly with input/output subsystems or devices. The programmer references all peripherals by use of file code designators (two alphanumeric characters), that are referenced in two ways: on file control records used by the allocator in GCOS software to specify those files needed to execute the activity, and in communicating with File and Record Control or the Input/Output Supervisor. The file code designators and their assigned Peripheral Assignment Table (PAT) are used by the Input/Output Supervisor for peripheral identification.

Series 200/2000 Compatibility Mode On Series 60 Systems

The Series 200/2000[1] Compatibility Mode on Series 60 systems extends the capabilities of the Series 60 systems by enabling the user to execute OS/2000 programs (and Mod 1 programs that can be run under OS/2000 control by use of the Mod 1 Simulator), and to process Series 2000 type data.

GCOS software schedules Series 2000 workloads in the same way that it schedules any Series 60 systems workload. Processing is done concurrently with other dimensions of GCOS software such as time sharing, batch, and remote job entry. Series 2000 object programs are run without recompilation. Series 2000 data is processed without conversion. The user need not rewrite the Series 2000 Job Control Language (JCL). This task is performed by a JCL translator.

[1]The term Series 2000, as used in this manual, encompasses both Series 20 and Series 2000 systems, including Model 820..

Series 2000 activities dynamically compete with all GCOS software activities for available system resources. Series 2000 activities can be mixed freely with standard GCOS activities to form a complete job. Any number of jobs that include Series 2000 activities can be run concurrently on Series 60 systems up to the maximum imposed by the installation.

The Series 2000 Compatibility Mode has three principal elements:

1. Compatibility Support Executive (CSE)

2. Series 2000 Object Program Image

3. Buffer Area

The Series 2000 processor obeys instructions in the object program image until it traps on a Monitor Call instruction which is a call for a supervisor function. The trap causes the CSE, a set of GCOS slave mode routines, to be entered to interpret the Monitor Call and to satisfy the function either within itself or by issuing GCOS supervisory calls. When the function is complete, CSE issues a dispatch request to GCOS software for the Series 2000 processor to continue obeying instructions in the object program image. CSE uses the buffer area in slave program memory in conjunction with Monitor Call requests for input/output operations.

An End-of-Program (EOP) Monitor Call in a Series 2000 program causes activity termination in GCOS software and an Unusual End-of-Program (UEP) Monitor Call results in an Abort condition in GCOS.

Series 2000 files enjoy all the advantages of the GCOS file system; for example, file protection and security (permissions and access control).

For further details of the hardware, software, and procedures that allow Series 2000 programs to be run on Series 60 systems, refer to the User's Guide to Series 200/2000 Compatibility Mode of Level 66 (CM66).

JOB FLOW

Batch job flow consists of the following phases:

- Input Media Conversion
- System Scheduler
- Allocation
- Execution
- Termination
- Output Media Conversion

The following paragraphs describe these phases and trace a hypothetical job as it goes through execution under GCOS software.

NOTE: Throughout this manual the terms job and activity have specialized meanings. An activity consists of a single program (such as FORTRAN compilation or an object program load and execution). A job is a set of activities which together constitute a problem or a production application. The activities constituting a job must be processed in sequence.

Input Media Conversion

When a job is presented to the system, either from an online card reader or any other input device (e.g., Front-End Network Processor, IMCV tape), it is processed by the System Input program. The System Input program separates the input for the job into two files: a control card file (J*) and a data card file (*J). The control card file describes the job and provides one-word pointer records to the various parts of the data file. The J* file eventually becomes the first part of the execution report. The data file is made up of various sub-files containing the data to be used by the job.

If remote batch input is presented to the system, the Remote System Input (RGIN) program processes input from the Network Processing Supervisor (NPS) or the General Remote Terminal System (GRTS). Incoming data blocks are buffered and stored in random access mass storage. Remote System Input also multiplexes job input from remote terminals connected to the Front-End Network Processor. Then input processing continues as with local job input.

The System Input program also has the capability to process job input consisting of a mixture of BCD, ASCII, and binary record input. ASCII input streams are accepted only from PSI-type (Peripheral Subsystem Interface) card readers, IMCV tapes, and mass storage devices. Control cards or job control language (JCL) must be in BCD, but ASCII (eight-bit character) data can be used in job data records bounded by $ ASCII and $ ENX JCL. The control cards can be mixed in with the ASCII data, but must be transmitted in BCD to the System Input program. When the System Input program detects the $ ASCII JCL, it starts writing the ASCII JCL records to a data file with a media code of 12 (octal). No special processing is performed. When a $ ENX, $ SNUMB, or ***EOF JCL is encountered, the ASCII mode is reset.

System Scheduler

Jobs processed by the System Input program are passed to the System Scheduler facility of GCOS software. This facility permits GCOS software to accept a large number of jobs and dispense them in a manner that permits efficient utilization of system resources. The maximum number of jobs that may be entered into the Scheduler depends on the amount of available file space. The System Scheduler also permits the user to tailor job streams to fit the requirements of his particular system.

Job priorities processed in the System Input program determine the handling of the jobs when they are passed to the System Scheduler. Jobs in the scheduler fall into one of the following classes:

o Express - Jobs that require minimum system resources are put in the express class. These are passed to the peripheral allocator for execution ahead of nonexpress jobs. The number of Express jobs that can be run at one time and the Express classification itself are defined by the user site manager.

o	Hold	- The Hold class contains jobs that the user has requested held by a $ MSG3 JCL statement until a specified date and time or until put into execution by the system operator. Additional cyclic scheduling parameters may be defined on the $ MSG3 JCL statement that will schedule a job at a specific time and thereafter reschedule the job at a fixed time interval.
o	Normal	- The user site manager can define 1-50 normal classes. Jobs placed in these classes are passed to the peripheral allocator for processing on a rotational basis by class: highest priority in the class and first in, first out. The Express class always gets priority in dispensing a job, and Normal class jobs get subsequent job dispensing on a rotating basis.

The System Scheduler also allows a user site to establish its own scheduling classes, through the use of the System Scheduler User Classification Module (.MSCAN). This facility is described in the System Operation Techniques reference manual.

Allocation

During allocation, peripheral devices and memory are assigned to a specific activity after the control tables are examined to determine gross peripheral requirements. Through this technique of allocating peripherals in advance of execution, the operator can perform functions such as mounting tapes required for a specific job, while other activities are in execution. It is during the allocation phase that mounting instructions are issued to the operator via the system console. When the assigned peripherals are ready and sufficient memory is assigned to the activity, the allocation phase is completed, and the activity passes into the execution phase.

If the peripheral requirements of an activity can be met only partially, the job is bypassed until all of its peripheral needs can be satisfied. The threshold value of 40 may be assigned by the operator if the allocation is not possible. This blocks peripheral and memory allocation for other activities, and other jobs are not considered for allocation until the requirements of this job can be met. No activity of a job is initiated until all prior activities of that job have been completed.

If specified by the user on a tape device card ($ TAPE, $ TAPE7, or $ TAPE9), magnetic tape units may be allocated according to class. The request for a specific class of tape unit may be satisfied by any nondedicated unit of a class equal to that specified or of a higher class than that specified. Device class structure must be defined at system startup.

Execution

During execution, each activity is executed under the supervision of the Dispatcher. All user communication with GCOS software is provided at execution time by the use of Master Mode Entry (MME) routines. These routines provide requests for service, exchange of information, and control functions. Many MME routines are called automatically by the compilers and by File and Record Control subroutines, and if using either of these, the user need not be concerned with a majority of the MME routines.

GCOS software features a dynamic dispatch queue. Program entries are in the queue only if they are ready to be executed. The queue is ordered according to priority of the programs to be executed. This priority may at installation option be based on a combination of job urgency and a ratio of channel time to processor time. The top priority program is at the head of the queue. Dispatching is therefore a simple and straightforward mechanism for picking the top entry from the queue and dispatching to it. Control is returned to the Dispatcher at the completion of interrupt processing.

Responses to any I/O exception conditions or processor faults are handled during this phase. When these are encountered, standard actions are performed by the system, or the programmer (or the operator) is given the option of handling the situation differently. This is done through Exception Processing routines (see Input/Output Programming reference manual).

Termination

A job or activity may terminate either normally or abnormally (see Section VI). The procedure followed in terminating a job or activity depends upon the type of termination. Upon program termination, an error and accounting record (see Section VII) is produced on the system output file (to be printed on the execution report), the system output file is closed, the operator is told if files require dismounting, tables used during execution are deleted, and memory storage and peripherals are deallocated.

When successive GMAP or FORTRAN activities are encountered without intervening $ IDENT or file control records, they are run as a single GMAP or FORTRAN activity. Thus a job containing ten FORTRAN compilations followed by a $ EXECUTE is run as two activities, a FORTRAN compilation of ten routines and an execution activity. Activity termination takes place only after the last FORTRAN compilation and after the execution activity.

Output Media Conversion (SYSOUT)

All output written on the System Output File is transcribed into printer or punched card output. The SYSOUT file may be output at the origin of the job, or at a specified remote station.

Example of Job Flow

The following sample job is traced through GCOS processing to illustrate the specific actions taken after the job is submitted in the form of the following JCL:

```
               1        8            16

               $        SNUMB        TST12,3
               $        IDENT        JOB12,JOE SMITH
Activity       $        OPTION       FORTRAN
   1           $        FORTRAN      LSTOU,DECK,DUMP
               $        INCODE       IBMF
                          .
                          .          Source Input
                          .
```

```
Activity     $       GMAP          LSTOU,DECK,DUMP
   2         $       TAPE          G*,X1D,,123,,ASSEMBLER

             $       EXECUTE       DUMP
             $       LIMITS        50,20K,4K,9K
             $       TAPE          02,X1R
             $       TAPE          03,X2R
Activity     $       TAPE          04,X3R
   3         $       INCODE        IBMF
                        .
                        .          Data
                        .
             $       ENDJOB
```

The deck TST12 is submitted to computer operations for processing.

INPUT MEDIA CONVERSION

Input Media Conversion scans the JCL for a $ SNUMB JCL statement, which tells the System Input program that a new job is beginning.

Activity 1.

Upon detecting the $ SNUMB JCL, System Input initializes the control stack for the first activity by placing the job identifier and urgency in their relative positions. The next card ($ IDENT JOB12,JOE SMITH) is found in the control stack buffer in its entirety. This information produces the error and accounting report for this activity. The next control card found is the $ OPTION card. System Input opens a Loader input (R*) file and puts the $ OPTION card onto this file. The $ OPTION FORTRAN card is used later during loading to set the standard load options for FORTRAN. The $ FORTRAN card indicates that the first activity has been defined. A $ SOURCE card is created and placed on the R* file; detail entries are made in the control stack for allocation of the B*, C* (deck option specified), K* (COMDK option specified), *1, P*, and S* files (see Section III) and the standard limits are set for the FORTRAN Compiler. The LSTOU and DECK options listed on the $ FORTRAN card are encoded into the switch word contained in the control stack buffer. These options override the standard options associated with the FORTRAN Compiler. Source input following the $ FORTRAN is written onto S* in the standard character set (encoded by the $ INCODE card).

Activity 2.

When the $ GMAP card is encountered, the standard GMAP Assembler limits are set up. The $ TAPE card creates a file entry for G*. (The $ TAPE card overrides the normal file which would be opened upon encountering the $ GMAP card without a following $ TAPE card.) The presence of the $ TAPE card signals the existence of source input on a tape named G* having reel number 123, which the operator must mount.

Activity 3.

Detection of the $ EXECUTE card signals that Activity 2 has been defined and the input for execution activity (Activity 3) is beginning. The limits specified on the $ LIMITS card override the standard limits normally used in an execute activity. In addition, when the $ EXECUTE card is detected, detail entries for the normal execution files I*, B*, and P* are made in the control stack, and bit 5 of the program switch word is turned on.

The next three $ TAPE cards make file detail entries for three tape files designated 02, 03, and 04, which will be assigned cyclically to a physical channel (X1, X2, and X3). Also in activity 3, the $ INCODE card signals the System Input program to transliterate data from the IBM FORTRAN character set to the standard character set. The data cards following the $ INCODE card are then placed on the I* file. The $ ENDJOB signals that this job has completed input media conversion and is now ready to be sent to the System Scheduler, which will dispense the job.

ALLOCATION, EXECUTION, AND TERMINATION

When the System Scheduler passes the job into allocation, control is returned to the Dispatcher. The Dispatcher enables the Allocator which checks the allocation urgency list. The Allocator reads the control stack for the first activity to determine the peripheral requirements. The peripheral devices and scratch files are allocated, and memory is allocated for the first activity. A MME GECALL is placed in the slave program prefix (32, octal) with a location specified to load FORTRAN from the software library, and compilation begins. Source input is read from the S* file, and the FORTRAN compilation is completed. The output generated is put on the output file (P*). Since the execution bit in the Program Switch Word was set, the output is also put on the B* file (for a subsequent "load and go" operation). The B* file is used by subsequent activities of this job, and therefore is saved and put into an abeyance table so that any future requests for B* are read from this file.

After termination of the first activity, control is returned to the Allocator, and allocation of the second activity begins. A MME GECALL GMAP is placed in the slave program prefix (32, octal) with a location specified to load GMAP from the software library, and the assembly begins. The GMAP Assembler reads source input from the G* file and puts its output on the B* file.

Allocation for the execution activity is enabled, and allocation of peripherals begins. Memory requested in the $ LIMITS card (20K) is added to the memory requirements for the General Loader (6K) minus any memory which can be shared with the General Loader during loading (4K). If sufficient memory is available, it is allocated. A MME GECALL for GELOAD is placed in the slave program prefix (32 octal) and execution begins. The object decks are read and loaded and then the program is entered at its primary entry point.

When execution (activity 3) is completed, all peripheral devices and memory are released, and the entries made in the allocation tables are deleted.

OUTPUT MEDIA CONVERSION

Output written on P*, C*, and K* files is printed or punched and the accounting record is produced.

SYSTEM SUMMARY

GCOS contains the following major functional components:

- o Startup
- o System Input
- o System Scheduler
- o Dispatcher
- o Peripheral Allocation
- o Rollcall, Memory Allocation, and Operator Interface
- o Fault Processing and Service MME's
- o I/O Supervision
- o Exception Processing
- o Termination
- o System Output
- o File System Maintenance
- o Utility Routines
- o Time Sharing

Each of these functions, which are summarized in the following paragraphs, is accomplished by one or more GCOS modules.

Startup

Startup is performed by a Startup program package which is, in itself, a miniature operating system. It provides a variety of system startup operations including loading, initialization, and utility functions required for system startup.

Startup permits a restart of the system from a mass storage peripheral device (disk) after a system fault. In this restart, the operator can specify new control sections of the Startup package (if any) to be re-input from the card reader or from the console via the Startup Console Editor (BCD card images only).

A startup option is available on the DPS 8 system to disable the hexadecimal floating point capability if all processors in the configuration do not have hex mode capability. To use this option the $ INFO JCL statement is used to ensure that the system is initialized with the hex mode off. The $ INFO statement is used as follows:

```
$ INFO HEXOFF
```

System Input

The System Input program modules accept system input from an online card reader, remote terminal, or other device. This input is then segregated into control and data files by these modules. System Input modules are called into memory only when needed, thus multiple input streams may be used.

System Scheduler

Jobs processed by the System Input program are passed to the System Scheduler. The System Scheduler uses the job priorities and classes processed in the System Input program to establish a dispensing queue to fit the requirements of the user's operation. The user site may also establish its own scheduling classes.

Dispatcher

The Dispatcher function keeps as many system components as possible in simultaneous use. It accomplishes this by selecting the highest priority program (either master or slave mode) that can make effective and immediate use of a processor and/or peripheral subsystems. This priority is based on options selected by the user site. The Dispatcher, thereby, allows overlapping of I/O operations of the various programs. A program, whether it is a GCOS system slave or a user slave activity, can be entered only from the Dispatcher. The Dispatcher resides permanently in main memory.

Peripheral Allocation

The peripheral allocation function is the primary activity scheduling function of the system. The peripheral allocator module, together with a group of modules that are called into memory when needed, schedules and allocates all peripheral devices used by the system. In this operation, the peripheral allocator issues the console instructions required for the allocation of devices and verifies device allocation before passing the job to the Core Allocator for memory allocation and execution.

System Rollcall

The system rollcall function, performed by the .MPOPM module at system startup, is a check of devices defined in the configuration section of the Startup deck. The status of each device is checked and devices not available for system operation are released. Released devices are identified on the system console so that they may be re-assigned later.

In the rollcall operation, special initialization procedures are performed for devices requiring them. When the rollcall function is completed, the rollcall portions of .MPOPM are erased from memory.

Memory Allocation

The memory allocation function, performed by the Core Allocator (modules that operate within the SSA of Program 01), allocates memory space to jobs entered into the system for processing. Memory space is allocated to a job based on job urgency (either specified in the job control cards or assigned by the Peripheral Allocator). Jobs are loaded into memory from the high address end. When a new job entering the system has no specific memory allocation requirements, it is allocated the smallest available memory area into which it can be loaded.

For high priority jobs, the Core Allocator performs memory compaction and job swapping to obtain the required memory space for the jobs. These modules also process memory requests associated with the MME GEMORE and MME GEMREL instructions.

Operator Interface

The .MPOPM module controls the operator-system message interface, except for those requests affected by the Program Number as a Resource option. Without the option installed, the messages exchanged between system programs and the system console are processed by these .MPOPM routines. With the option installed, messages are processed from the module issuing the message running in $ PALC.

Fault Processing

The method by which faults are processed depends upon whether the fault occurred in GCOS software or in a slave program.

On a GCOS software caused fault, an error message is sent to the system console, and the contents of the registers (at the time of the fault) are dumped. If a noncontrol processor found the fault, it requests an interrupt from the control processor. The control processor puts the noncontrol processors in a DIS (Delay until Interrupt Signal) state and takes the dump. If the control processor detects the fault, it puts the noncontrol processors in a DIS state and takes the dump.

If a program fault is encountered when operating in the slave mode, the fault causes interrogation of the user fault vectors. If the user specifies a fault processing subroutine, control is transferred via this vector back to the slave activity. If no fault processing subroutine is specified in the user fault vector, a MME GEBORT is set in the user program and control is transferred to MME GEBORT (see Section IV). GCOS allows the following faults to be processed optionally by a user error subroutine.

Program Faults

1. Memory	5. Illegal Procedure
2. Divide Check	6. Fault Tag
3. Overflow	7. Derail
4. Command	

A user can process an unlimited number of Divide Check, Overflow (including those caused by truncation condition), and Derail faults. However, except under special conditions, a second attempt to process a Fault Tag, Command, Illegal Procedure, or Memory fault causes the program to abort. The special condition is that the instruction counters and indicators (IC and I) (even) word of the fault vector pair is zero and the second word is nonzero. Under this condition, after the first fault is processed, GCOS software allows the user to process one of these faults.

The following faults do not allow the user to go to a fault processing routine.

<u>Hardware Faults</u>

Lockup	Timer Runout
Parity	Shutdown
Op Not Complete	Execute
Startup	Connect

In addition to those faults listed above, the Master Mode Entry (MME) instructions are also processed as faults, under control of the Fault Processor. The MME instructions give the user programmer the ability to call the master mode routines needed for the execution of the program.

<u>I/O Supervisor</u>

I/O Supervisor (IOS) performs the acceptance, initiation, and termination of all I/O requests. I/O operations on the system are accomplished by using a MME GEINOS instruction. These instructions, followed by pertinent I/O parameters, give control to IOS to initiate activity on the peripheral subsystems.

In addition, IOS provides the following capabilities:

- o Symbolic-to-Physical Unit Translation. Programmers refer to files symbolically (file codes) in the various source languages available within the software system. This enables maximum flexibility of assignment of files at execution time, because the files which a program may use are device independent. The I/O Supervisor converts the symbolic file designations to absolute physical assignments at execution time.

- o Simulated Tape Processing on Disk. IOS provides a method of data processing on disk which simulates the serial mode of processing normally associated with magnetic tape. This mode of processing (linked) not only provides some degree of device independence to the computer configuration, but also allows the opportunity to reduce program setup time by eliminating the use of magnetic tape for some files.

- o Supervision of Interrupts. IOS responds to all I/O interrupts and takes appropriate action. The programmer will normally not be required to concern himself with the interrupt system of the computer.

- Queueing of I/O Requests. IOS maintains queues of I/O commands. Many commands can be queued for each I/O device. In multidevice channel subsystems, such as magnetic tape and disk, IOS will not necessarily select queued commands for execution in the order in which they were requested. However, for a linked file, the commands are executed in the requested order.

- Utilization of IOS with Crossbarred Magnetic Tape Subsystems. If the primary channel is busy at the time I/O is to be initiated, IOS automatically selects an alternate channel (if available) on a crossbarred magnetic tape subsystem.

- Pseudo Command. In the special case of the typewriter, a pseudo command is provided which allows a programmer to type a message requiring an answer. Using this command guarantees that the answer to a question will follow in sequence behind its asking.

- Awareness. The I/O Multiplexer (IOM) provides information to a processor which allows IOS to maintain awareness of the status of each peripheral device. IOS maintains tables which allow it to record such information as a magnetic tape in rewinding, a peripheral device in Ready status, or a device in Standby status.

- File System Protection. IOS verifies that the user has been granted permission for the permanent file access.

- Accounting. IOS keeps a complete record of the time spent by the processor and each peripheral for every program executed. These statistics are later recorded on the Statistical Collection File (SCF) and printed on the execution report for use in billing and system operation analysis. The user interface with IOS is defined in the Input/Output Programming reference manual.

System Output

Output media conversion is accomplished through SYSOUT. Data is stored on system storage and handled by output media conversion with SYSOUT. As soon as the data is transferred to the output media, the system storage is available for reuse as storage for additional data directed to the output destination. The storage space is not released until needed so that reprint/repunch capabilities are available.

PAGE PRINTER SYSTEM OUTPUT

The Page Printer Executive (PPE) handles the dispersion of all Page Printer System (PPS) directed output from system output space in either the online or off-line (stored on tape) mode.

The PPE is spawned from the Level 66 console (OPNSUTIL file). Initialization includes processing of a site prepared options file. One of the options is a site specified PPS control block file for default usage. Either this file or a PPS-software file establishes the default PPS control block. An initiated PPE remains inactive until one of the following requests is received:

- A Level 66 console request to create a PPS format tape for off-line printing.

- An on-line request for output by the Level 6/43 PPS application via DAC interface.

GCOS File System

The heart of GCOS software is a centralized file system of hierarchical tree-structured design. This design provides multiprocessor access to a common data base, file protection, and maximum access control. The file system is composed of a permanent online data base controlled by a master catalog which identifies each user. Within the master catalog are pointers to catalogs for every user. Each user's catalog may name and point to still lower level catalogs.

The File Management Supervisor (FMS) provides automatic allocation and protection of secondary storage space. The user need have no concern as to which device or type of device a given file is on. The file system provides:

- o Selective access control
- o Security protection
- o Hierarchical catalog/file structures
- o Communication between the batch, remote batch, and time sharing dimensions of GCOS.

These features are optional. If they are not needed, the file system allows a user to proceed in a simple, straight-forward manner in regard to them. The user may, on the other hand, utilize the file system in as sophisticated a fashion as system operations dictate.

The common data base storage allows a truly integrated multimode system, since the several processing modes not only coexist, but also communicate with each other. One application of this capability might be to allow a large batch job to generate or update a file (perhaps based upon other files entered from time sharing terminals) and have that file available for inquiry by time sharing users.

Another FMS application allows a time sharing user to generate a job for batch processing. The user program in the batch mode may be too large to execute conveniently in the time sharing mode, or may be an existing program for which modification for direct execution is not desirable.

FMS provides a simple, high-level means of defining physical file space and controlling access to it. As such, it has nothing to do with file content. Actual input/output is performed via the standard I/O facilities provided by GCOS; e.g., File and Record Control and GMAP. Thus, creating a file refers to using the FMS to define a specific extent of space. This space is thereby associated with a given file name and file creator.

There are two modes of user communication with FMS. Batch and remote batch users communicate via the FILSYS activity language. Time Sharing System users at remote terminals communicate by means of the ACCESS subsystem in the Time Sharing System.

Honeywell Error Analysis And Logging System

The Honeywell Error Analysis and Logging System (HEALS) is an integral part of GCOS software. It is designed to ensure system availability and data integrity on the system. The key components of HEALS are:

- Processor instruction retry by software. HEALS Instruction Retry automatically retries most Operation Not Complete (ONC), Interface Parity and Cache Memory Parity (PAR), and Illegal Procedure (IPR) faults.

- Logging, analysis, and reporting of processor faults. This function includes the following capabilities:

  1. Console reporting of Instruction Retry activities.

  2. Retention of processor error records on a HEALS file.

  3. Logging, analysis, and automatic report printing of hardware errors in the processor/main-store subsystem.

- Enabling, re-enabling, and monitoring of cache memory. HEALS enables and re-enables cache memory. It retries cache memory parity errors and logs the occurrence of cache memory errors.

- Monitoring of MOS EDAC-connected errors. HEALS monitors MOS memory EDAC-connected errors, and reports their occurrence.

- Logging, analysis, and reporting of MPC statistics. This function includes the following capabilities:

  1. Retention of MPC statistics on a HEALS file.

  2. Error rate monitoring. An error threshold message is output if an adjustable error rate threshold is exceeded.

  3. Printing of MPC statistics from data retained in the controller or from the HEALS file.

- Logging and reporting of Exception Processing statistics. This feature allows collection and reporting of Type 03 Exception Processing error records.

Associated with HEALS is a System Abort History subsystem that collects and retains information on how, where, and how often system aborts occur. The reports produced by this subsystem are affiliated with the hardware error reports produced by HEALS.

REDUCED PERIPHERAL CONFIGURATIONS

A System may be tapeless, cardless, printerless or any combination thereof. Startup evaluates the device type codes of peripherals configured on the system during Startup and defines in the communication region option word (.CROPT) the absence of card readers, printers and/or tape drives.

The image of the Startup deck used for bootloading the system resides on disk for tapeless system and, since no card reader may be available on the system changes to the startup deck (during bootload) can be accomplished via the console Startup Editor. Editing the startup deck by this method permits individual sites to modify the Startup deck to meet specific requirements as needed. For detailed operating information refer to System Operating Techniques reference manual.

REMOTE MAINTENANCE CONSOLE

The RMC66 is a logical console device which resides on the remote port of a Low Cost Console (LCC) channel adapter. This logical console consists of a Diagnostic Processor Unit (DPU) with its on-site console and remote port which has a dial-in capability for technical assistance center (TAC) usage. The LCC remote port and DPU are proprietary hardware limited to use by the Field Engineering Representative (FER). Characteristics of the RMC66 are listed below:

o It is an optional Console

o Primary purpose and function is to support Field Engineering maintenance.

o It is accessible by one or two primary channels.

o It will change from or to a ready state during system Startup and/or operation.

o It is not a candidate for automatic reassignment of system console I/O. The operator can, however, move system console traffic to or from the RMC66.

o Software accessibility is limited to System programs including test and diagnostic programs and user programs with operator permission.

The remote maintenance console messages are defined in System Console Messages reference manual.

GCOS SOFTWARE SUPPORT OF EIS PROCESSORS

GCOS software allocates a 32-word buffer to each program to serve as a storage area for Extended Instruction Set[1] (EIS) registers. One set of 16 words saves the registers across program interruptions. The additional set of 16 words saves the main level EIS registers during a courtesy call.

---

[1]Extended Instruction Set (EIS) refers to an extension to the original hardware instruction set. EIS is the standard instruction repertoire for Models 6025, 6040, 6060 and 6080 of the Series 6000 system and all models of the Series 60 Level 66 system.

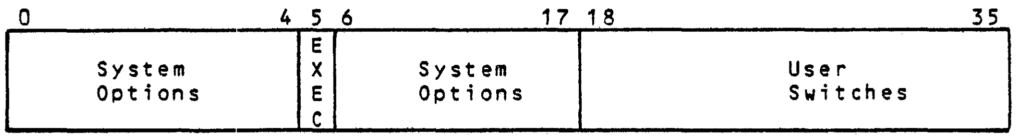
Normally the EIS register buffer for ordinary slave programs resides in the second SSA at location 775740 relative to the program's lower address limit (LAL).

The EIS register buffers for privileged programs reside in main memory. The communication region symbol .CREIS points to a table that contains the locations of all currently active EIS register buffers, ordered by program number.

SECTION II

PROGRAM SWITCH WORD

The Program Switch Word contains job switches, system options, and user switches, as shown below.



Bit 5 denotes whether there is a following execution activity within the job. If an execution activity follows, bit 5 = 1; if none, bit 5 = 0. Bit 5 is used by GCOS to control program processing and cannot be altered by the user.

PROGRAM SWITCH WORD SYSTEM OPTIONS

System options are stored in bits 0-4 and 6-17 of the Program Switch Word. The status of the bits is retained only while the activity is in execution. When the option is used, the associated bit is set to a 1. The options available with each software processor system are described in the Control Cards reference manual. The options and the associated bits controlling them are given below.

Bit	Bit=1	Bit=0
0	DUMP Option	NDUMP Option
1	FORTRAN, GMAP or PL/I	Neither FORTRAN, GMAP, nor PL/I
2	SYMTAB/XREF	NSYMTAB/NXREF
3	PURGE implicit file	NPURGE
4	CLEAR memory	NCLEAR
5	Execute activity follows	No execution activity follows
6	COMDK/ON1	NCOMDK
7	DECK/ON2	NDECK
8	LSTOU/ON3	NLSTOU
9	UPDATE/ON4	No UPDATE
10	LSTIN/ON5	NLSTIN
11	DEBUG, NGMAC, STAB/ON6	NSTAB, GMAC, NDEBUG
12	(Bit 1=0) Abort Subactivity	(Bit 1=0) No Abort Subactivity
	(Bit 1=1) MAP	(Bit 1=1) NOMAP
13	GMAP/System Editor Interface	No GMAP Interface
14	(Bit 1=0) COPY/FORTRAN or COBOL created G* File	(Bit 1=0) NCOPY/standard G*
	(Bit 1=1) FORM	(Bit 1=1) NFORM
15	(Bit 1=0) GESAVE/PRODUCT	(Bit 1=0) No GESAVE/PRODUCT
	(Bit 1=1) LNO	(Bit 1=1) NLNO
16	(Bit 1=0) EXTEND	(Bit 1=0) No EXTEND
	(Bit 1=1) BCD	(Bit 1=1) ASCII
17	(Bit 1=0) Reserved	(Bit 1=0) Reserved
	(Bit 1=1) OPTZ	(Bit 1=1) NOPTZ

These bits are set as each activity is allocated. Bits 0-2 and 6-11 are set after encountering a language processor control card ($ COBOL, $ GMAP, $ IDS, etc.) or a $ EXECUTE control card specifying options.

USER SWITCHES

The user switches (bits 18-35) are retained between activities of a job and are available for use by the programmer for whatever need he may have within his job. The user sets the switches by means of a MME GESETS or a MME GERETS (see Section IV).

STRANGER OPTIONS

The Stranger Option File (D*), which is created by the ALC1 module of the Peripheral Allocator, accommodates language processor and loader options which cannot be represented by bits in the user switch word. These options are transparent to GCOS software and must be interpreted by the user program. This capability is permitted by the conditional use of the D* file.

The D* file is a Read/Write mass store file, up to one llink (320 words) long in standard system format. If the program call card contains options in the variable field which are not represented by bits in the user switch word, the card image is written to the D* file by ALC1. The card image on the D* file is truncated; i.e., trailing words of blanks are deleted so that the logical record is 14 words or less.

Any $ ETC card associated with a program call card specifying stranger options is also copied on the D* file. If no stranger options are encountered on the program call card then only the $ ETC card will appear on the D* file.

No selective editing is performed by the Peripheral Allocator, thus any standard options (those represented by bits in the user switch word) can be included in these card images. However, the appropriate switch word bits for the standard options will be set by the allocator.

The use of the D* file by the user overrides the use of this file by the allocator. The user may overwrite the D* file up to a maximum of 320 words. The D* file created by ALC1 may not be saved between activities.

SECTION III

SYSTEM FILES

System files are created by GCOS software for its own use and/or for use by software processor systems (GMAP, FORTRAN, General Loader, etc.). System files used as input to other software processors working under GCOS software are created by the System Input program. In addition, files are sometimes created by this program and passed between software processors; e.g., as in FORTRAN compilation. In this case, the program creates S*, which is used as input to FORTRAN, which then generates the object code for the activity.

The system files used with each processor system are listed in Figure 3-1.

System files and user files are designated symbolically by two characters. The system files are distinguished from user files in that one of the two characters is an asterisk. For example, A* and *1 refer to system files, whereas AB and C1 can be used to refer to user files. Like user files, most system files are in standard system format. Standard system format is defined in the File and Record Control reference manual.

Included in this section are descriptions of the following system files:

- Alter file (A*)
- Object program file (B*)
- Simulator input file (*B)
- Binary deck file (C*)
- Stranger option file (D*)
- GMAP source file (G*)
- Program link file (H*)
- Data storage file (I*)
- Compressed deck file (K*)
- System library file (L*)
- Secondary subroutine library file (*L)
- System output file (P*)
- Loader input file (R*)
- Compiler source file (S*)
- I-D-S source file (*S)
- Utility file (U*)
- SYSEDIT file (X*)
- Intermediate file (*1 - *7)

System files are normally allocated space on the primary (faster) system storage device. If space is not available on the primary device, the files are allocated space on the next faster system storage device.

Software Processor System	Files Used	Size	Type
ALGOL	A*	Variable	Linked
	B*	2	Linked
	C*	-	
	K*	-	
	P*	-	
	S*	Variable	Linked
	*1	4	Linked
COBOL	A*	Variable	Linked
	C*	-	
	G*	4	Linked
	K*	-	
	P*	-	
	S*	Variable	Linked
	*1	4	Linked
	*2	1	Linked
	*3	15	Random
	*6	10	Random
		Plus files allocated for GMAP	
FORTRAN	A*	Variable	Linked
	B*	2	Linked
	C*	-	
	K*	-	
	P*	-	
	S*	Variable	Linked
	*1	4	Linked
General Loader	B*	Variable	Linked
	H*	5	Random
	L*	-	
	*L	-	
	P*	-	
	R*	Variable	Linked
GMAP	A*	Variable	Linked
	B*	2	Linked
	C*	-	
	G*	Variable	Linked
	K*	-	
	P*	-	
	*1	4	Linked
I-D-S	A*	Variable	Linked
	C*	-	
	K*	-	
	P*	-	
	S*	6	Linked
	*S	Variable	Linked
	*1	4	Linked
	*3	15	Random
		Plus files allocated for COBOL.	

Figure 3-1. Files Used by Software Processors

Software Processor System	Files Used	Size	Type
JOVIAL	A*	Variable	Linked
	B*	2	Linked
	C*	-	
	K*	-	
	P*	-	
	S*	Variable	Linked
	*1	4	Linked
	*2	1	Linked
Utility	P*	-	
	U*		
DATANET 355/6600 Simulator (355SIM)	*B	2	Linked
	P*	-	
	SV	-	
Source File Editor	F*	1	Linked
x	J*	Variable	Linked
x	M*	Variable	Linked
	*Z	5	Random
x	*K	Variable	Linked
x	K*	Variable	Linked
	A*	3	Linked
x	*C	Variable	Linked
	B*	5	Linked
	*1	4	Linked
	G*	4	Linked
	S*	6	Linked
x	*2	Variable	Linked
x	*3	Variable	Random
	C*	-	
	P*	-	
	*5	3	Linked
x	*7	Variable	Linked
x	4*	Variable	Linked
x	5*	Variable	Random
Object File Editor	F*	1	Linked
x	J*	Variable	Linked
	B*	Variable	Linked
x	*R	Variable	Linked
x	R*	Variable	Linked
	*4	1	Linked
	*5	3	Linked
x	*7	Variable	Linked
	P*	-	

These System Input-created files use as many links as are necessary to contain the input data involved.

These files, which are used as intermediate data files, are initially allocated a minimum number of links, as shown. That number may be increased dynamically via MME GEMORE during their use by the various language processors.

NOTE: x - Files used by the Source and Object Libraries but not allocated by GCOS software. Editor files are fully described in the Source and Object Library Editor reference manual.

Figure 3-1 (cont). Files Used by Software Processors

ALTER FILE (A*)

The alter file (A*) is created by the System Input program and used by GMAP, ALGOL, COBOL, I-D-S, JOVIAL, and FORTRAN. Upon reading a $ UPDATE card, this program puts the $ ALTER cards following that $ UPDATE card into the A* file. The file is then used to insert the appropriate changes into the program being assembled or compiled. The A* file consists of the $ ALTER control cards and source or symbolic language cards which are to be merged with the cards of the primary input file. A detailed description of the alter procedure may be found in the File and Record Control reference manual.

OBJECT PROGRAM FILE (B*)

The object program file (B*) is created by GMAP, JOVIAL, FORTRAN or ALGOL, and used by the General Loader whenever a $ EXECUTE control card is included in the source deck and no fatal error has occurred during compilation. When the General Loader requires the object deck of the subprogram or program compiled, it loads the object program from the B* file rather than from the R* file, which the loader normally reads for its input.

During normal allocation, the System Input program allocates two links for the B* file. This initial allocation may be increased dynamically via MME GEMORE during the use of B* file by the various language processors.

SIMULATOR INPUT FILE (*B)

The simulator input file (*B) is used by the DATANET 355/6600 Simulator program to contain the Simulator program control cards and binary deck. It is also used to simulate card reader input.

BINARY DECK FILE (C*)

The binary deck file (C*) contains the binary object deck created by the GMAP Assembler, ALGOL, COBOL, and/or FORTRAN, etc. The file is written for punching in the following format:

```
$ OBJECT
      .
      .        Object Deck
      .
$ DKEND
```

The C* File is created when the deck option is specified in the appropriate control card ($ GMAP...DECK, $ FORTRAN...DECK, etc.).

The file medium is SYSOUT and the file is limited by SYSOUT. The report code (bits 30 to 35) of the record control word should be 76 (octal). (Refer to File and Record Control reference manual.)

GMAP SOURCE FILE (G*)

The GMAP source file (G*) is created by System Input program or COBOL, and is used by GMAP. The System Input program creates the G* File when encountering a $ GMAP control card. The COBOL compiler creates a G* File as its output file. The G* File is then used as input to the GMAP Assembler.

Initially, G* is allocated four links; however, the size may be increased dynamically (via MME GEMORE) by the various language processors. When used as input to GMAP, the size of G* varies and is dependent upon the number of links required to contain the input data.

PROGRAM LINK FILE (H*)

The program link file (H*) is created by the allocator and utilized by the user program for temporary storage of programs that are called into and out of memory. Procedures used to create an H* File are described in the General Loader reference manual. Loading the subsequent links contained on H* is accomplished by using the LLINK or LINK subroutine, as described in the FORTRAN Subroutine Libraries reference manual. Restarting a link job from a saved H* tape is described in the Service Routines manual (RLHS program).

The General Loader writes programs onto the H* file in the order in which they are overlayed and not in the order that they are read into memory. The first two words of a linked program contain (1) the link entry point and the location of the link's DEBUG table, if used and (2) the upper and lower addresses of memory.

Unless intended for reuse, the H* file is usually written on disk and has a size of five links if a $ LINK card is present.

DATA STORAGE FILE (I*)

The data storage file (I*) is created by the System Input program and used by the various user programs.

COMPRESSED DECK FILE (K*)

The compressed deck file (K*) is created by the GMAP Assembler or any of the system language processors. The file created is a compressed deck image of the source program processed by the language processor, including the merged alter file input. The K* file is created when the COMDK option is specified in the appropriate control card (that is, $ GMAP....COMDK). This file with changes or corrections can then be used by the appropriate assembler or compiler as its input. A detailed description of the compressed deck format is provided in the File and Record Control reference manual. The K* file is normally assigned to SYSOUT. However it can be assigned to tape, disk, or punched cards by using the appropriate control card (that is, $ TAPE....K*, $ FILE....K* or $ PUNCH....K*). A detailed description of the control cards is given in the Control Cards reference manual.

SYSTEM SUBROUTINE LIBRARY FILE (L*)

The system subroutine library file (L*), when created by the object library editor, is used by the General Loader as a serial file. It may also be created as a random file library by the program RANLIB.

The L* file contains the system subroutine library. These subroutines are in relocatable binary card image records and are loaded by the General Loader. The subroutines include the FORTRAN I/O, math routines, File and Record Control, etc.

SECONDARY SUBROUTINE LIBRARY FILE (*L)

When space requirements of the subroutine library demand, the *L file may be used to contain the less frequently used subroutines. The *L file is assumed to reside on a slower device and is scanned before the primary file (L*).

SYSTEM OUTPUT FILE (P*)

The system output file (P*) is created by the various software systems (GMAP, General Loader, ALGOL, etc.) to collect all system output. The P* file is normally assigned to SYSOUT, which has a line limit determined by the type of activity. However, if the limit of SYSOUT may be exceeded, the P* file can be assigned to tape or disk. The output can then be printed by means of the Bulk Media Conversion (BMC) program, which has no such limits.

LOADER INPUT FILE (R*)

The loader input file (R*) is created by the System Input program and used by the General Loader. The System Input program places all loader control cards and their associated object programs onto the R* file. When a language processor activity is specified by a control card ($ GMAP, $ ALGOL, etc.), a $ SOURCE card is generated and placed on the R* file. This is in anticipation of loading the newly assembled program from the object program file (B*) in a compile-and-go type of job.

The Loader reads the pertinent control cards and loads the object program from either the R* file or the B* file (the B* file is used if the object program was reassembled).

COMPILER SOURCE FILE (S*)

The compiler source file (S*) is created by the System Input program and used by COBOL, ALGOL, JOVIAL, and FORTRAN. Upon reading a $ COBOL, $ ALGOL, etc., System Input creates the S* file, which contains all the source cards following the program call card. When the compilation is about to be run, the S* file is read and used as source input to the appropriate compiler.

I-D-S SOURCE FILE (*S)

The I-D-S source file (*S) is created by the System Input program and used to contain I-D-S/COBOL DATA statements. The file is used as input to the I-D-S translator, which puts the statements into a form acceptable to the standard COBOL compiler.

UTILITY FILE (U*)

The utility file (U*) is created by the System Input program for use with the Utility Routine. The U* file contains the control cards and data for Utility. It may be assigned to and read from magnetic tape by means of a $ TAPE control card.

UTL2 FILE (*U)

The UTL2 file (*U) is created by the System Input program for use with UTL2. The *U file contains the control statement and data for UTL2. It may be assigned to and read from magnetic tape by means of a $ TAPE control statement.

SYSEDIT FILE (X*)

The X* file is provided by the user for use in a SYSEDIT activity during a total system edit.

INTERMEDIATE FILE (*1)

The intermediate file (*1) is created by FORTRAN, I-D-S, JOVIAL, COBOL, ALGOL or GMAP and is used by FORTRAN, COBOL, I-D-S, ALGOL, or GMAP for working storage. If the number of links initially allocated to this file (4, 8 or 12 depending on the source file size) is exceeded during the Write phase, additional links will be allocated as necessary as long as links are available on the system storage device (see MME GEMORE, Section IV).

INTERMEDIATE FILE (*2)

The intermediate file (*2) is created by JOVIAL, COBOL, or I-D-S and is used by JOVIAL or COBOL for working storage. If the number of links initially allocated to this file is exceeded during the Write phase, additional links will be allocated as necessary as long as links are available on the system storage device (see MME GEMORE, Section IV).

It is initially allocated one link for COBOL or JOVIAL programs.

INTERMEDIATE FILE (*3)

The intermediate file (*3) is created by the COBOL compiler and used by I-D-S and COBOL for working storage. Its allocated space is 15 random links. If this limit is exceeded, the user program is aborted. If required, the user may allocate more space; up to 105 links.

INTERMEDIATE FILE (*4)

The intermediate file (*4) is used by the Object Library Editor to contain information pertinent to the Record of Changes file. The *4 file is initially assigned one linked link on system storage.

INTERMEDIATE FILE (*5)

The intermediate file (*5) is used to contain all editor execution reports, to keep processor execution reports and editor execution reports separated. At the end of the Source and Object Library Editor run, the contents of *5 are transferred to P*. These reports may be kept separate by assigning *5 to SYSOUT. Normally, the *5 file is initially assigned three linked links on system storage.

INTERMEDIATE FILE (*6)

The intermediate file (*6) is used by COBOL as a working file.

INTERMEDIATE FILE (*7)

The intermediate file (*7) is used by the Editor to contain source programs and/or object decks generated via the PUNCH directive. The *7 file must be assigned by the user. It may be assigned to tape or mass storage to be used as input to the Bulk Media Conversion (BMC) program for punch card output. For small jobs, *7 can be assigned to SYSOUT. The *7 file cannot be assigned to a punch device.

STRANGER OPTION FILE (D*)

The stranger option file (D*) is created by the Peripheral Allocator to accommodate stranger options specified in the program call card. Stranger options are options which are not represented by bits in the user switch word. When stranger options are detected in the variable field of the program call card, the image of the card, and any related $ ETC cards, will be copied onto the D* file.

The D* file is a Read/Write mass store file one llink (320 words) in length in standard system format. The D* is associated only with the activity which created it and may not be saved between activities.

NOTE: File codes M0-M5 are restricted to I-D-S and GCOS software use.

SECTION IV

MASTER MODE ENTRY ROUTINES

MME OPERATIONS

In the multiprocessing, multiprogramming environment of the Series 60 and Series 6000 systems, user programs cannot execute master mode instructions. However, the Master Mode Entry (MME) capability in GCOS software permits user programs to call on master mode routines required in the execution of their program.

When a MME instruction is executed, the processor (either control or noncontrol) places itself in master mode and control is given to the Fault Processor, which processes the MME instructions. The Fault Processor stores the program registers, instruction counter and indicators (to allow reference and return to the requesting program). It then examines the address portion of the MME instruction, which contains a numerical value corresponding to the symbol used by the programmer. The processor then transfers control to the pertinent MME routine. A nonvalid address in the MME instruction causes the program to be aborted. Some MME instructions are not allowed during courtesy call.

Figure 4-1 defines each mnemonic symbol for the MME instruction, the symbol meaning, and the associated value substituted in the MME address field by the assembler. The text in this section describes most MME operations in detail. However, some MME operations are directly related to an individual software system, and in order to present them in the proper context, these MME's are covered in the appropriate software manuals. In this case, the MME entry in this section contains a reference to the appropriate software manual. The MME definitions in this section are arranged in alphabetical order by operation name.

Before a return is made to the user program, the program registers are restored to their original values, except for those MME's which return information to the user in registers.

Although mass storage allocation is in llinks (a llink = 320 words), MME mass storage parameters must be expressed in links. The GCOS processor associated with the MME converts the link parameter to llinks before requesting allocation of mass storage.

MME	Footnote	Definition	Value Octal	Value Decimal
.EMM	a	Enter Master Mode	36	30
GEBORT	a	Request to abort program	10	8
GECALL	c	System loader	22	18
GECHEK	c	Checkpoint Dump	27	23
GEENDC	a	End Courtesy Call	16	14
GEFADD	a	Physical file address request	3	3
GEFCON	a	File control block request	12	10
GEFILS	a	File switching request	13	11
GEFINI	a	Terminal transfer to monitor	7	7
GEFRCE	c	ISP Abort Processing	40	32
GEFSYE	c	File System	41	33
GEIDSE	b	I-D-S	35	29
GEINFO	a	Information GCOS entry	45	37
GEINOS	a	Input/output request	1	1
GELAPS	a	Elapsed time request	6	6
GELBAR	c	Load Base Address Register (BAR)	37	31
GELOOP	c	Loop protection	33	27
GEMODER	a	Disable HEX mode	54	44
GEMODES	a	Enable HEX mode	53	43
GEMORE	b	Request for additional peripherals or memory	11	9
GEMREL	b	Release part of memory	25	21
GENEWS	e	Spawn new job	43	35
GEPRIO	a	I/O priority	42	34
GERELC	a	Relinquish control	17	15
GERELS	b	Component release	4	4
GERETS	c	Reset switch request	15	13
GEROAD	b	Roadblock	2	2
GEROLL	c	Reinitiate or rollback program	31	25
GEROUT	a	Remote output record	30	24
GERSTR	c	Read file in system format	24	20
GESAVE	c	Write file in system format	23	19
GESECR	f	Password encryption	46	38
GESETS	c	Set switch request	14	12
GESNAP	d	Snapshot dump request	5	5
GESNUM	a	Supply SNUMB number	44	36
GESPEC	a	Special Interrupt Request	20	16
GESYOT	c	Write on SYSOUT	26	22
GETIME	a	Date and time-of-day request	21	17
GEUSER	c	User-supplied MME (depends on user code)	32	26
GEWAKE	c	Call-me-later	34	28
GEXLIT	a	Transliteration Table Acquisition	47	39

Footnotes:

a Works correctly in Courtesy Calls.
b Immediate abort in Courtesy Call.
c Not checked in MME Processor and may not work correctly in Courtesy Call.
d Aborts MME in slave mode. Does not abort in master mode, but may not work correctly.
e Aborts MME only. .CALL allowed in Courtesy Call.
f Privity MME only.

General - Any MME process that causes a Roadblock is illegal within a Courtesy Call.

Figure 4-1. MME Symbols And Definitions

.EMM -- ENTER MASTER MODE

This MME is used to request entry into master mode operation. An attempt by a user to execute this MME results in an abort of the user program unless the program has been granted special master mode privileges. The operator performs this function when an authorized job contains a $ PRIVITY JCL statement.

Entry is made to this routine from the fault vector as a result of the MME .EMM.

Calling sequence is:

```
L       MME .EMM
L + 1 return
```

On return, all registers have been preserved except index registers 5, 6, and 7 which are set to the values from .CRLAL, the program number and the executing processor number, respectively.

Attempted use of the entry without privity results in a slave program abort for reason 'Illegal MME address'.

It is the programmer's responsibility to follow standard coding conventions in master mode.

GEBORT -- REQUEST TO ABORT PROGRAM

MME GEBORT is used to specify abnormal termination of an activity. Memory allocated to the activity is released, all peripherals allocated to the activity are deallocated, and dismounting instructions are issued where necessary.

A memory dump is written on the execution report if bit 0 of the program switch word is set. If the address (bits 0-17) of word 27 (octal) of the Slave Program Prefix is within the slave program limits, return is made to the user for wrapup (see Section VI).

Entry to this routine is made from the fault vector as the result of the MME GEBORT.

The calling sequence for this routine is as follows:

```
L      MME     GEBORT

Initially: C(Q) required

               Bits  0-23  Ignored
                    24-35  Two-character reason code
```

The choice of reason code is at the discretion of the user. Neither character can be 77 (77 is a special command to the printer).

All I/O activity should be terminated prior to executing this master mode entry or it may be lost.

GECALL -- SYSTEM LOADER

MME GECALL makes an entry in the input list of the appropriate service program, enables the execution of that program, and prevents return to the requesting program until the user's request has been successfully carried out.

Entry to this routine is made from the fault vector as the result of a MME GECALL.

The calling sequence is as follows:

```
L      MME     GECALL
L+1    BCI     1,XXXXXX
L+2    ZERO    Location of first word, Error return address
L+3    ZERO    Transfer address
```

The system program identified by XXXXXX in L+1 is loaded into memory at the location specified in bits 0-17 of L+2. If this field is zero, the data will be located at the location specified in word 5 of the control block. If an error occurs during the CALL service, the Call module will return to the address specified in bits 18-35 of L+2 (if zero, the program is aborted). If Call service has no error, return is to the location specified in L+3. If L+3 is 0, processor control is transferred to the entry point of the system program when loading terminates.

On normal return, the A- and Q-registers contain the following:

A-register:	Bits	0-17	Loading origin
		18-35	Number of words
Q-register:	Bits	0-17	Transfer address
		18-35	Not used

On an error return, bits 18-35 of the Q-register contain one of the following error codes:

Code	Definition
52	I/O limits error
53	I/O error
54	No PAT for call file
55	Bad device call
56	Nonrandom GECALL file
60	Call checksum
63	Call name missing
64	Call out of file span
65	Improper call

The error return address field is zeroed upon return to prevent looping. The error return address field must be reset by the caller if successive calls are desired.

MME GECALL should not be used during courtesy call.

Checksum Verification Option

Checksum verification on module reads or GECALL loads is optional. The CHKSUM option of the Startup $ INFO control statement is used to instruct the system to perform the verification. If the CHKSUM option is not present, no verification is done.

GECHEK -- CHECKPOINT DUMP

MME GECHEK initiates a checkpoint dump and sets up bookkeeping to enable the program requesting the MME GECHEK to be later rolled back (via MME GEROLL) to the last checkpoint taken. The use of MME GECHEK and MME GEROLL are described in detail in the Program Recovery/Restart manual.

The calling sequence for MME GECHEK is as follows:

```
L       MME     GECHEK
L+1     TRA     1,QU
L+2     Next instruction
```

For programs using File and Record Control, the Q-register must contain the following on entry to MME GECHEK:

```
Bits   0-17 = Location of the file control block for the checkpoint file
      18-19 = Zero
         20 = 1, If this is a pre-GERELS checkpoint
      21-35 = Zero
```

For programs that do not use File and Record Control, the Q-register must contain the following on entry to MME GECHEK:

```
Bits    0-5    Must be zero
       6-17    File code of the checkpoint file
         18 = 0
         19 = 1
         20 = 1, If this is a pre-GERELS checkpoint
      21-35    Must be zero
```

In either case, the file must have write permission. Also, if bit 20 = 1, the A-register, bits (0-17), must contain the address of the MME GERELS list (see below).

Additional Calling Sequence Options

Each checkpoint dump contains sufficient information for restarting the activity or job (via MME GEROLL instruction) at a later time. The restart is made from the point at which the checkpoint was taken. The normal return from a MME GEROLL is to L+1 of the MME GECHEK calling sequence with QU = the address +1 of the MME GECHEK. A user also has the option of providing an additional routine, if special slave processing is required as a result of a rollback. After a rollback this routine is automatically entered if, prior to the first MME GECHEK, the entry location is placed in the upper half of word 14 (decimal) in the user's memory area. When using this option, the user is responsible for re-entering the main body of the program where needed. The upper half of the Q-register will still contain the address +1 of the most recent MME GECHEK.

If a slave program wants to take a checkpoint and immediately follow it by a total release of some data files and/or peripherals, the slave program must set bit 20 = 1 in the Q-register when the MME GECHEK is executed. Also the A-register, bits 0-17, must contain the address of the MME GERELS list. A maximum of 38 files can be listed. The MME GERELS must be issued immediately following the MME GECHEK. When a MME GEROLL is done, these files will not have to be present and will not be restored in the SSA.

The format of the list pointed to by AU is as follows:

```
ZERO     n,0
BCI      1,0000F1
BCI      1,0000F2
  .
  .
  .
BCI      1,0000Fn
```

n above equals the number of files being released and F1, F2, ... Fn are the file codes of the files being released.

General Rules

Any number of checkpoints may be written on a checkpoint file. The last complete checkpoint is the only one available for tape rollback however. After switching tapes on a multireel file, the File and Record Control program provides an automatic checkpoint dump on the user's checkpoint file if the upper half of SPA word 14 is non-zero (indicating that the user has taken a previous checkpoint).

MME GECHEK can be used to take a checkpoint dump (on permanent mass storage file only) followed by a job termination. Subsequently, a MME GEROLL can be used to restart the job at the checkpoint. This is called job suspension with deferred restart. (See further discussion under MME GEROLL.)

A MME GECHEK should not be used during a courtesy call.

Neither the restart procedure nor the rollback procedure provides file positioning capabilities for files assigned to card readers, card punches, or printers.

Users of FMS ACCESS/MONITOR/ are not permitted to take consecutive checkpoints without an intervening MME GEFSYE to release befores whenever one or more pages have been reserved between the checkpoints. The following sequence is suggested:

```
Read/Writes
MME GECHEK
MME GEFSYE
Read/Writes
MME GECHEK
MME GEFSYE
  .
  .
  .
```

Other sequences are allowed (and any sequence is allowed in non-concurrent access), but the above provides continuous protection in the case of concurrent or single access.

After the successful release of a complete file, MME GERELS forces the Checkpoint bit OFF (in .CRSN1) and the No Activity Restart bit ON except in the case of a pre-GERELS checkpoint. This prevents activity restart or rollback until a new checkpoint is taken. The checkpoint indicator is not turned off by a MME GERELS to change dispositions or access mode. Any attempt to use MME GERELS to change the disposition of a protected file causes the program to abort with an INVALID MME GERELS REQST (octal code 167) abort message.

On successful allocation of a new file, MME GEMORE forces the Checkpoint bit OFF, and a new checkpoint is required for activity restart or rollback. The checkpoint indicator is not affected by a MME GEMORE to grow a file.

Checkpoint dumps may be directed either to a separate file or to the controlling data file itself. The latter alternative may be elected only if the data file resides on magnetic tape, is opened only for output, and is closed without the LOCK option.

Dual Checkpoints

If a mass storage checkpoint file is big enough to accommodate two checkpoint dumps, then MME GECHEK will alternate writing checkpoints to the first and last halves of the file. This feature protects against an abort occurring during a checkpoint since the last complete checkpoint can always be used by rollback.

MME GECHEK Error Returns

If an error was encountered during the MME GECHEK processing, no checkpoint is taken. An error code is returned to the slave program in the lower half of the Q-register.

The octal error codes are as follows:

1 -	File specified for checkpoint not open. No checkpoint taken.
2 -	No Write permission on checkpoint file or file not open for output.
3 -	Error in positioning of checkpoint file.
4 -	Invalid checkpoint file device.
5 -	Bad status return on attempted write. An incomplete checkpoint dump may exist and a new checkpoint should be taken.
6 -	Error encountered in pre-GERELS information (A-register address out-of-bounds, too many entries in list, incorrect format of list).
7 -	End-of-Tape encountered on checkpoint file.
10 -	Two checkpoints were taken in a job using concurrent access files and befores were not released between checkpoints.
11 -	Checksum error.
12 -	MME GECHEK not allowed in courtesy call.
13 -	The checkpoint file was not large enough to contain a checkpoint dump.
14 -	Error in saving file position.

GEENDC -- END COURTESY CALL

MME GEENDC terminates the courtesy call that was in progress.

Entry to this routine is made from the fault vector as result of a MME GEENDC.

The calling sequence is as follows:

```
L      MME    GEENDC
```

The program is aborted if MME GEENDC is executed outside a courtesy call.

GEFADD -- PHYSICAL FILE ADDRESS REQUEST

The MME GEFADD is used to obtain physical addresses, disposition codes, and specified file information (for mass storage) for use in operator messages. This routine searches the user program Peripheral Assignment Table (PAT) for the requesting program, looking for an entry containing the specified file code.

If a match is found for the file code supplied in the Q-register, the A-register is set to a nonzero value. Return is to the requesting program with the physical device address in bits 6-23 of the Q-register and with the file attributes defined in the A-register.

If no match was found in the PAT, both the A-register and Q-register contain zero. The use of the A-register distinguishes between an unsuccessful search and a successful search in which the I/O controller channel and device addresses are legitimately all zero.

Entry to this routine is made from the fault vector as the result of executing the MME GEFADD instruction. At the time of executing the MME, bits 24-35 of the Q-register must contain the pertinent file code. The calling sequence for this routine is as follows:

```
L       MME     GEFADD
L+1     Return
```

Initially: C(Q) required -

Bits	0-23	Must be zero
	24-35	Two-character file code

Results: C(Q) - For all devices

Bits	0-5	Zero
	6-11	Device address
	12-13	IOM number
	14-17	Four-bit channel address
	18-23	Six-bit channel address
	24-35	Reserved for GCOS

C(A) - For card readers and card punches

Bits	0-5	Device code of primary channel SCT
	6-8	Disposition code as follows: 000 Release 001 Dismount 010 Save 011 Continue
	9-10	= 00, Device does not have ASCII capability. = 11, Device has ASCII capability.
	11-12	Reserved for GCOS
	13-14	For CRZ301 and CRU1050 defines 51-column card option as follows: 10 51-column card option active on this device 11 51-column card option available on this device For other card reader and card punches, these bits are reserved.
	15-35	Reserved for GCOS

C(A) - For Printers

Bits	0-5	Device code of primary channel SCT
	6-8	Disposition codes as follows: 000 Release 001 Dismount 010 Save 011 Continue
	9	= 0, Device does not have ASCII capability. = 1, Device has ASCII capability, but this capability may not be enabled, depending on the print train loaded.
	10	= 0, ASCII capability not currently enabled on this device. = 1, ASCII capability exists and is currently enabled on this device. This bit set if ASCII print train is loaded.
	11	Must be zero
	12-14	Octal value defining print line capabilities: 000 Reserved 001 132-character print line 010 136-character print line 011 160-character print line (PRT401, PRT402, PRU1200, or PRU1600 printers only) 100-111 Reserved for GCOS
	15-35	Reserved for GCOS

For card readers, card punches, and printers, bits 9-14 may define the type of device and the device capability as follows:

Bits 9-14(Octal)	Device	Capability
00	CPZ201	BCD
60	CPZ300	ASCII capability
60	PCU0120/PCU0121	ASCII capability
60	CPZ301	ASCII capability
00	CRZ201	BCD
60	CCU0401	ASCII capability reading and punching
60	CRZ301/CRU1050 /CRU0501	ASCII capability
62	CRZ301/CRU1050	51-column card option active
63	CRZ301/CRU1050	51-column card option available
02	PRT201	BCD,132 print positions
01	PRT203/PRU1100	BCD,132 print positions
42/62	PRT300	ASCII capability,136 print positions
42/62	PRT303	ASCII capability,136 print positions
42/43/62/63	PRT401/PRU1200	ASCII capability,136 or 160 print positions
42/43/62/63	PRT402/PRU1600	ASCII capability,136 or 160 print positions

C(A) - For mass storage devices

Bits	0-5	Device code of primary channel SCT
	6-8	Disposition code as follows: 000 Released 001 Dismount 010 Save 011 Continue
	9-17	Reserved for GCOS
	18	= 0 Links defined in bits 22-35 = 1 Llinks defined in bits 22-35
	19	= 0 Sequential (linked) file = 1 Random file
	20	= 0 Temporary file = 1 Permanent file
	21	= 0 Physical block size = 64 words
	22-35	Number of links or llinks depending on bit 18 (low order 14 bits)

C(A) - For magnetic tape devices

Bits	0-5	Device code of primary channel SCT (bits 0-5 of primary SCT)
	6-8	Disposition code as follows: 000 Released 001 Dismount 010 Save 011 Continue
	9-12	Density capability from bits 21-24 of word 1 of secondary SCT.
	13-14	For a magnetic tape subsystem to be used in 2000/60 Magnetic Tape Interchange Facility, these are defined as follows: 10 Allocated tape unit capable of S2000 mode of operation 11 tape file allocated as S2000 file
	15-17	Reserved for GCOS
	18	MTS0610
	19	Reserved for GCOS
	20-23	High-density default from .CROPT (data originates in $ INFO card in Startup deck)
	24-27	Low density default from .CROPT
	28-31	Density from bits 0-3 of word 3 of the Peripheral Allocation Table (PAT)
	32	Mode indicator for 1600-bpi tape subsystem
	33-34	Density history bits
	35	Reserved for GCOS

C(AQ) =0 if file released or system device.

User programs should not use physical addresses in I/O commands. IOS ignores physical device addresses specified by a user program in an I/O command and inserts the proper physical address corresponding to the file code.

No file code match will be found if the specified file code pertains to a file that is designated on a control card as a SYSOUT or remote file. The A- and Q-registers contain all zeros if the file is not present or is a system device.

The mode indicator (bit 32 of A-register) and the density history bits (bits 33-34 of A-register) are defined as follows for 1600-bpi tape subsystems:

Mode Indicator	Density Bits	Device Density
1	00	6250 bpi
1	11	1600 bpi
0	00	556 bpi
0	01	800 bpi
0	10	200 bpi

For ASA subsystems, these are defined as:

Mode Indicator	Density Bits	Device Density
1	01	High Density
0	00	Low Density

The mode indicator bit is the same as the device high density/low density bit (bit 18) of the word 2 of the secondary SCT for the device. The density history bits are the same as bits 13-14 of word 1 of the secondary SCT.

The tape density code is defined as follows:

0000	As is (no mapping)	
0001	200 bpi	NRZI mode
0010	556 bpi	NRZI mode
0100	800 bpi	NRZI mode
1001	1600 bpi	Phase-Encoded (PE) mode
1100	6250 bpi	
1111	System high density and low density options used.	

For a mass storage device, MME GEFADD returns a file size in bits 22-35 of the A-register if the file size is equal to or less than 16,383 llinks. If the file size is larger than this, the MME GEFADD will return a zero in these bits of the A-register.

If a file may be larger than 16,383 llinks, the user must define the file size via MME GEFCON or MME GEFSYE. For a structured catalog file, the user can use a MME GEFSYE with function code 33 (see File Management Supervisor reference manual). If the file is on a non-structured, non-cataloged device, MME GEFCON must be used.

GEFCON -- FILE CONTROL BLOCK REQUEST

MME GEFCON places the following information, identified by the Q-register, in the File Control Block (FCB).

- o Link/random bit (provided for disk)
- o SYSOUT indicator
- o Type of device
- o Physical device address (same format as bits 6-23 of MME GEFADD)
- o File serial number (provided only when the file is defined as magnetic tape)
- o Reel sequence number (provided only when the file is defined as magnetic tape)
- o File present indicator
- o C or D disposition indicator (provided for magnetic tape)
- o 7- or 9-track bit if file is magnetic tape

When additional file control blocks are linked to the original file control block (via LOCSYM-1 of the FCB), the above information is supplied for all of the files in the chain. The chain is ended by LOCSYM-1 being zero or pointing to the beginning of the chain. This is not the chain of opened files defined by slave prefix location 17 (octal), but the list of files being processed by a single call to OPEN.

An alternate function of MME GEFCON is to obtain a magnetic tape reel number from an opened FCB (at LOCSYM-7) and insert it in the PAT (word 4) for this file so that it may be output for accounting purposes.

Entry to this routine is made from the fault vector as the result of the MME GEFCON.

The calling sequence for this routine is as follows:

```
L       MME     GEFCON
L+1     Return
```

Initially: C(Q) required

```
        Bits  0-17  Location of first FCB
                18  = 0 Normal use of MME GEFCON by File and
                        Record Control.
                    = 1 File serial number in bits 0-29 of FCB-7
                        is to be placed in PAT entry word 4
                        (bits 6-35).
                19  = 0 No density to be specified.
                    = 1 Tape density replaced in PAT entry word 4
                        (bits 0-3).

             20-31  Reserved.

             32-35  Tape density (MTS500 and MTU0400/MTU0500)
                    as follows:

                    0000 - As is
                    0001 - 200 bpi
                    0010 - 556 bpi
                    0100 - 800 bpi
                    1001 - 1600 bpi
                    1100 - 6250 bpi
                    1111 - High/low defaults

    C(FCB-4)

        Bits  0-23  Reserved
             24-35  Two-character file code.

    C(FCB-7)

        Bits  0-29   File serial number
             30-35   Reserved for GCOS
```

The results of a MME GEFCON instruction depend on the type of device.

1. For magnetic tape devices, if a file serial number is present in PAT entry word 3 (bits 6-35), that number is stored in FCB-7. Bit 18 of PAT entry word 2 is tested to determine whether or not a serial number is present. If no serial number is present, 999990 is stored in FCB-7.

   The reel sequence number from bits 27-35 of PAT entry word 2 is stored in bits 27-35 of FCB-8. If the reel serial number is zero, a 1 is stored in these locations.

   In FCB-0, bit 24 = 0 indicates seven-track tape; =1 indicates nine-track tape.

   The deallocation code from bit 23 of the PAT pointer (nonSYSOUT) is stored in bit 29 of FCB-5.

2. For SYSOUT devices, bit 25 of FCB-0 is set if the output file is assigned to a SYSOUT device.

3. For remote devices, bit 23 of FCB-0 is set if the output file is assigned to a remote terminal. Device code 5 is stored in bits 26-29 of FCB-0.

4. For mass storage devices, bit 8 of PAT entry word 2 is stored in bit 24 of FCB-0. This bit defines file type as follows:

   0 - Linked
   1 - Random

   The file size in links or llinks contained in bits 0-17 of PAT entry word 3 is stored in FCB-7 as follows:

   a. If the file size is equal to or less than one link (12 llinks), the value is stored in both upper (bits 0-17) and lower (bits 18-35) halves of FCB-7. For example:

      FCB-7 = 000014000014

   b. If the file size is greater than one link the number of llinks in the last link is contained in bits 0-17, while bits 18-35 contain the number of links minus one plus the number of llinks in the last llink. For example:

FCB-7 = 000014000014	One-link file
FCB-7 = 000001000002	13-llink file
FCB-7 = 000014000015	Two-link file

   If the file size exceeds 16,383 llinks, bit 0 of FCB-7 is set to 1 and the number of llinks is contained in bits 14-35.

5. For direct access devices, the device type from bits 6-11 of the PAT pointer is stored in bits 26-29 of FCB-0, and the unit designator is stored in bits 24-29 of FCB-1.

6. For all devices, if the file is present, bit 18 of FCB-5 is set, and the device type code is stored in bits 26-29 of FCB-0. The permissible octal device codes are as follows:

   2 - Magnetic tape
   5 - Remote device
   6 - mass storage device
   7 - Conversational device
   10 - Card reader
   12 - Printer
   14 - Perforated tape
   16 - Card punch

For complete details on use of the file control block, refer to <u>File and Record Control</u> reference manual.

GEFILS -- FILE SWITCHING REQUEST

MME GEFILS is used for switching primary and secondary logical units (magnetic tape devices). The secondary logical unit for this file code becomes the primary logical unit, and the primary logical unit, in turn, becomes the secondary logical unit. If no secondary unit has been assigned, the peripheral device assigned to the primary logical unit is assumed. The physical device address of the resultant primary logical unit is returned in the Q-register.

Entry to this routine is made from the fault vector as the result of a MME GEFILS.

The calling sequence for this MME is as follows:

```
L      MME     GEFILS
L+1    Return

Initially:  C(Q) required

       Bits  0-23  Must be zero
            24-35  Two-character file code


Results:    C(Q)

       Bits  0-5   Zero
             5-11  Device address
            12-13  IOM number
            14-17  Four-bit channel address
            18-23  Six-bit channel address
            24-35  Reserved for GCOS


     C(A)

       Bits  0-5   Device code (same as in bits 0-5 of word 1
                     of primary channel SCT)
             6-8   Disposition code as follows:
                       000  Released
                       001  Dismount
                       010  Save
                       011  Continue
             9-19  Reserved for GCOS
            20-23  High density default from .CROPT
            24-27  Low density default from .CROPT
            28-31  Density from bits 0-3, word 3 of PAT
               32  Mode indicator for 1600-bpi tape subsystem
            33-34  Density history bits
               35  Reserved for GCOS
```

If the file is released or is a system file, the A- and Q-register contain zeros.

The mode indicator and the density history bits are defined as follows:

	Mode Indicator	Density Bits	Device Density
	1	00	6250 bpi
	1	11	1600 bpi
	0	00	556 bpi
	0	01	800 bpi
	0	10	200 bpi
ASA	1	01	High Density
Subsystem	0	00	Low Density

The mode indicator bit is the same as the device high and low density bit (bit 18) in word 2 of the secondary SCT for the device. The density history bits are the same as bits 13-14 of word 1 of the secondary SCT.

The tape density code is defined as follows:

0000	As is (no mapping)	
0001	200 bpi	NRZI mode
0010	556 bpi	
0100	800 bpi	
1001	1600 bpi	Phase-Encoded (PE) mode
1100	6250 bpi	
1111	System high and low density options are used	

When files must be switched, it is the user's responsibility to issue any mounting or dismounting instructions to the operator. When using the File and Record Control function, the mounting and dismounting instructions are automatically issued. When purging files, files with alternate reels, which have executed a reel switch through the use of MME GEFILS or File and Record Control call to FORCE, will not be purged. Multireel files using a single tape drive will have only the current reel purged.

GEFINI -- TERMINAL TRANSFER TO MONITOR

MME GEFINI is the standard method for bringing an activity to a successful completion, thus allowing the next activity in the job to be allocated. Memory allocated to this activity is released, and all peripherals allocated to this activity, except those which have been designated as save, are deallocated. Dismounting instructions are issued if necessary.

Entry to this routine is made from the fault vector as the result of the MME GEFINI.

The calling sequence for this routine is as follows:

```
L       MME     GEFINI
```

All I/O activity should be terminated prior to executing this master mode entry or it may be lost.

GEFRCE -- ISP AND LABEL ABORT PROCESSING

When the Indexed Sequential Processor (ISP) or Label processor needs to abort a user program that called it, MME GEFRCE is used to process the abort. This MME routine produces the error message for the aborted program's execution report and aborts the program. The abort processing permits ISP or Label to produce a meaningful description of the abort condition followed by a standard ISP termination message.

Entry to this routine is made from the fault vector as the result of the MME GEFRCE.

The calling sequence for the Label routine (module LBL5) is as follows:

```
L     MME GEFRCE
L+1   ZERO (+), Positive Value, Abort Message Number
```

The calling sequence for the ISP routine (module ISPA) is as follows:

```
L     MME   GEFRCE
L+1   ZERO  -1, Abort Message Number
L+2   ZERO  Parameter List Pointer, Number of Pointers in List
```

The -1 indicates that the MME GEFRCE is being used by ISP. A negative function code other than -1 in this field produces the following message:

```
INVALID FUNCTION CODE SPECIFIED FOR MME GEFRCE
```

The parameter list pointer specifies the location of a list of message parameter pointers. The lower half of this word contains the number of words in this list. The list is organized in the following format:

```
B     ZERO     Parameter Pointer, Number of Characters
```

If the value for the number of characters is positive, the parameter in the location specified by the parameter pointer is assumed to be in BCD format, and the first n characters (where n = number of characters) from that location are moved into the abort message that is to be printed on the execution report.

If the value for the number of characters is negative, the parameter at the specified location is assumed to be a binary integer. This integer is converted to a six-digit BCD integer and the rightmost n digits (where n = number of characters) are moved into the abort message.

The abort messages are contained in the ISP abort module (ISPA). Each message is followed by a control word containing 0-3 compressed tally words that identify the positions within the message to which parameters can be moved and the maximum allowable parameter size. A MME GEFRCE call may specify fewer parameters than the maximum. However, if it specifies more than the maximum, the excess parameters are ignored. Similarly, a parameter size in the MME GEFRCE list may be smaller than the number of characters permitted by the message control word, but if the parameter is larger only the number of characters permitted by the message control word are moved. If an invalid message number is specified, MME GEFRCE produces the following message:

```
UNDEFINED ERROR MESSAGE NUMBER nnnnnn
```

If a location outside the user program is specified in the MME GEFRCE calling sequence, the following message is produced.

```
INVALID 'MME GEFRCE' PARAMETER ADDRESS
```

GEFSYE -- FILE SYSTEM ENTRY

User programs can call on the File Management Supervisor to provide file services in the execution of the programs. This call for service is accomplished by a MME GEFSYE. The function code in the MME GEFSYE defines the service to be performed. Use of MME GEFSYE is described in the File Management Supervisor reference manual.

The generalized calling sequence for MME GEFSYE is as follows:

```
L       MME GEFSYE
L+1     Courtesy Call or 0, Argument List
L+2     Function code, Buffer
L+3     Return
```

The function codes (in decimal) and their corresponding cataloging and protection service requests are as follows:

Cataloging

```
 2 = Catalog create
 3 = File create
 9 = File delete (purge)
10 = Catalog modify
11 = File modify
22 = File delete (release)
23 = File query
28 = Security lock
29 = Abort lock
38 = TDS lock
```

Protection

```
30 = Read from duplicate set and reset
30 = Reserve pages
30 = Specify TDS functions
34 = Identify defective space
35 = Replace and/or mark defective space
40 = Cancel changes
41 = Regard changes as complete
43 = Provide allocated file information
44 = User supplied befores
46 = File query for allocated file
```

GEIDSE -- JOURNALIZATION AND SUBFILE PAGE RANGE

MME GEIDSE is restricted to use by the I-D-S object time subroutines.

GEINFO -- INFORMATION ACCESS ENTRY

MME GEINFO allows a slave program to access specific information outside the program's normal Base Address Register (BAR) limits. Either or both of the following functions may be requested:

1. Information from the upper Slave Service Area (SSA) of the user program can be copied to a buffer provided by the user. The system macros should be used to access the copied SSA data as illustrated in the following example.

2. A List Pointer Word (LPW) can be supplied to allow one or more single units of information to be placed in the slave program's main memory. The LPW is processed by an SSA module.

Entry to this routine is made from the fault vector as the result of MME GEINFO instruction. The calling sequence for the routine is as follows:

```
L     MME      GEINFO
L+1   ZERO     Buffer end, Number of words
L+2   ZERO     LPW, Size of list
L+3   Return
```

```
Where:  Buffer end       - Address +1 of the end of a buffer
                           storage area
        Number of words  - Size of storage area
        LPW              - Address of first word of list of
                           directives to be processed
        Size of list     - Number of words of list directives
```

```
NOTE:  If L+1 contains zeros, the SSA copy function is not performed.
       If L+2 contains zeros, the LPW function is not performed.
```

The format of the list of directives pointed to by L+2 is as follows:

```
                    N - ZERO  Address, Option
                        ZERO  Address, Option
                         .       .       .
                         .       .       .
     N+No. of words-1-  ZERO  Address, Option
```

Address is the location at which the requested data word is to be stored. Option is the option number as follows:

OPTION	Address of word which contains:
1	Processor time remaining (copy of .SALT)
2	SYSOUT lines remaining (calculated from .SSYOT 0-17 - .SSYOT 18-35)
3	Channel time remaining (copy of .SACHT)
4	Processor time used (copy of .SPRT)
5	SYSOUT lines used (from .SSYOT 18-35)
6	Channel time used (copy of .STCHT)
7	Time-of-day start of activity (copy of .START)
8	Program state word (copy of .STATE)
9	Job urgency (from .SURG 18-23 aligned to bits 30-35)
10	Configuration data (copy of .CRFIG)
11	BCD time in hours-minutes-seconds (formatted as hhmmss)
12	Program number from SNUMB
13	Software release number (copy of .CRSR)
14	System default number of lines per printer page (copy of .CRPSZ)
15	Julian date (copy of .CRJCD)
16	Shared system number (.CRSSN)
17	Startup option word (copy of .CROPT)

Option 12 permits the use of MME GEINFO to obtain the program number for a specified SNUMB. For example, the Interslave Communication facility (INTERCOM) may require a program number for its MME GEINOS operation. MME GEINFO with option 12 can be used to obtain the number. In this example, the location of a SNUMB of $TRAX (left justified in bits 0-29) is entered into the location specified in the address field, and the program number, if found, is returned in bits 30-35. If no program number is found, bits 30-35 are set to zeros, but bits 0-29 are not altered.

Option 17 permits the use of MME GEINFO to determine which FORTRAN compiler is resident in the system. If BIT 3 is ON, option 17 indicates DML FORTRAN.

The following deck setup illustrates a MME GEINFO call.

```
1      8        16
-------------------------------------------------------
       LODST    .G3SYM       To define symbols
       MME      GEINFO
       ZERO     AAA,.LSST
       ZERO     BBB,2
       LDA      AAA+.SMSZ    Effective address of .SMSZ
                             in the memory buffer AAA
         .
         .
         .
AAA    BFS      .LSST
BBB    ZERO     EEE,2
       ZERO     GGG,5
EEE    BSS      1
GGG    BSS      1
```

```
Where:  EEE - Contains remaining SYSOUT lines
        GGG - Contains SYSOUT lines used
```

If the size of the buffer area equals .LSST (upper SSA length), the entire contents of the upper SSA is transferred. If the buffer size is less than .LSST, only an upper segment of the SSA is transferred. For example, if the buffer size is 2, only .SUID is transferred.

GEINOS -- INPUT/OUTPUT REQUEST

MME GEINOS is used for all user program I/O requests (see Input/Output Programming reference manual). The master mode routine that processes the MME GEINOS takes one of the following actions:

1. It aborts the program if any of its select sequence parameters refer to memory locations outside the program limits.

2. It aborts the program if the file identification (file code) is illegal for the requesting program.

3. It postpones acceptance of the I/O request if the pertinent request queue is already in use.

4. It queues the request, initiating I/O action immediately, if the channel is not busy.

5. It erases all data from the point at which the tape is positioned to the end-of-tape when the erase command is received.

The Interslave Communication (INTERCOM) facility is a special type of MME GEINOS, which may be used only in slave mode. It permits two or more in-memory programs to communicate directly through a buffer-to-buffer exchange of data. The INTERCOM facility uses a three-word calling sequence.

In general, a request for I/O is made by means of the MME GEINOS instruction followed by a three-word or five-word select sequence, depending on the number of peripheral commands required. (The five-word sequence is used for dual command MME GEINOS instructions; e.g., Seek-Read, Seek-Write, etc.).

Entry to this routine is made from the fault vector as the result of the MME GEINOS. The calling sequences for this MME are as follows:

The three-word calling sequence is:

```
MME GEINOS
Operation word
Identification word
Return word
```

The five-word calling sequence is:

```
MME GEINOS
Operation word 1
Identification word 1
Operation word 2
Identification word 2
Return word
```

Certain I/O requests involve two related operations which must be performed sequentially; for example, SEEK/READ (or WRITE) on the disk and WRITE/READ on the console.

The identification word of the calling sequence points to a file control word that can be used to specify a Relinquish Control (MME GERELC) or Roadblock (MME GEROAD) as part of the MME GEINOS operation. Bits 18-19 of the file code word can contain a 0 for a MME GEINOS operation, a 1 for a MME GEINOS with a Relinquish Control immediately following the I/O, or a 2 for a MME GEINOS with a Roadblock following the I/O.

GELAPS -- ELAPSED TIME REQUEST

MME GELAPS provides the requesting program a report of the total amount of processor time the program expended up to the time of the request.

Entry to this routine is made from the fault vector as the result of a MME GELAPS. Return is to the requesting program with the elapsed processor time right-justified in the Q-register. Time is expressed in number of 1/64 millisecond increments.

The calling sequence for MME GELAPS is as follows:

```
L      MME     GELAPS
L+1    Return
```

Results: C(Q) describes elapsed processor time

The amount of processor time charged to a program is affected by the following factors:

1. Competition by an I/O controller and/or another processor for available memory cycles.

2. Housekeeping inaccuracies unavoidably introduced when a processor responds to a peripheral interrupt or time runout fault. Consequently, the use of MME GELAPS is not recommended for timing studies.

GELBAR -- LOAD BASE ADDRESS REGISTER

MME GELBAR allows a slave program to reset the base address register to a smaller area within his allocated memory.

Entry to this routine is made from the fault vector as a result of MME GELBAR.

The calling sequence for this routine is as follows:

```
L      MME      GELBAR
```

Initially: C(Q) describes processor time increment before interrupt.

The processor time increment must be less than the amount of time remaining before the timer runs out. The time was assigned when the slave program was allocated. If the increment is greater, its value is set to the remaining allocated time.

Initially: C(A) required:

```
Bits  0-17   LOC1
     18-35   Not used
```

LOC1 and LOC1+1 contain the following:

```
LOC1   - Bits  0-17 - Base Address Register (BAR)
              18-35 - LOC2

LOC1+1 - Bits  0-17 - IC
              18-35 - I
```

LOC2 is the location of the program registers stored prior to issuing MME GELBAR (NOTE: LOC2 must be a modulo 8 address). The BAR is the desired Base Address Register setting relative to slave zero. The Instruction Counter (IC) is relative to the new base where execution is to be continued. The Indicator (I) equals the setting desired.

At the time of execution of MME GELBAR, the contents of the Q-register are stored in word 21 (octal), and the new BAR is stored in word 31 (octal). In addition, bits 19 and 30-35 of word 31 (octal) are set to zero, and the value actually loaded into the timer register is stored in word 22 (octal).

When any fault, timer runout, or return after having serviced any system interrupt occurs, GCOS returns to word 23 (octal) of the slave program after restoring registers with their contents at the time of the interruption. If word 23 (octal) is zero, the program is terminated.

Before returning to cell 23 (octal), GCOS puts the IC and I at the time of the interruption in word 22 (octal) and subtracts the amount of time used in GELBAR from word 21 (octal). In addition, bit 19 is set to 1 and the fault type is placed in bits 30-35 of word 31 (octal) if the interruption was a fault or timer runout. The time in word 21 (octal), upon return, can be negative.

The MME GELBAR should not be used during courtesy call.

GELOOP -- LOOP PROTECTION

MME GELOOP provides slave program protection by dividing the activity execution time limits into segments. This is particularly useful during program debugging when a program may get into a loop and remain in the loop until the limits specified on the $ LIMITS control card are exceeded. A timeout in a MME GELOOP normally results in an abort of the activity in execution with an I8-RUN TIME EXHAUSTED message even though the activity run time allowed by the system has not been exhausted. When the MME GELOOP timeout does not exhaust the activity run time and the user has specified a return address, control is returned to the specified address for subsequent processing within the same activity. The desired return address must be placed, by the programmer, in the lower half (bits 18-35) of word 27 (octal) of the slave program prefix.

If a return address is specified, registers at the time of dispatch to this return address will be as they were when the timeout occurred. If the user wants registers preserved, it is his responsibility to save and restore them.

Entry to this routine is made through the fault vector as a result of a MME GELOOP in a user program.

The calling sequence for this routine is as follows:

```
L       MME     GELOOP
L+1     Return
```

Initially: C(Q) = Requested loop time (in seconds), right justified.

The user can specify an infinite time (Dec -1). In this case, the total time will be the total activity limit specified in the $ LIMITS control card. Under no circumstances can the time on the $ LIMITS card be exceeded by use of the MME GELOOP.

The MME GELOOP should not be used in a courtesy call.

GMODES/GMODER HEXADECIMAL FLOATING POINT

Two MME's are provided to enable and disable the hexadecimal floating-point mode (HEX mode) in the batch processing environment. Once HEX mode is enabled, bit 32 of the Indicator Register must be set to initiate HEX mode processing. These MME's and their decimal values are:

GMODES	enable HEX mode	43
GMODER	disable HEX mode	44

Bit zero of the Q register must be set to enable or disable the HEX mode. The calling sequence is:

SET

```
     LDQ  =0400000,DU         Set bit 0 of Q register
     MME  GMODES
     CMPQ =0400000,DU         Is HEX enabled?
     TZE  (HEX available)          YES
           or
     CMPQ =0400000,DU
     TNZ  (HEX not available)       NO
```

RESET

```
     LDQ  =0400000,DU
     MME  GMODER
```

If the MME GMODES is executed to enable the HEX option, on return, bit zero of the Q register is set to a one state. If the HEX option is not available on the system, bit zero of the Q register is set to a zero state.

If MME GMODES successfully enables the HEX option, bit 8 of the .STAT1 is set to a one state. Subsequently, GMODER resets bit 8 of .STAT1 to a zero state.

At the start of each activity within a job .STAT1 is initialized (reset), therefore, the HEX mode must be enabled in each activity. Bit 33 of the mode register is set and reset in conformance with bit 8 of .STAT1. In other words when a program is dispatched and bit 8 of .STAT1 is set, bit 33 of the mode register is also set. Conversely, when the program is taken out of execution, bit 33 of the mode register is reset (zero state) and bit 8 of .STAT1 is reset to zero.

GEMORE -- REQUEST FOR ADDITIONAL PERIPHERALS OR MEMORY

MME GEMORE is used to obtain an additional tape handler, additional links (3840-word blocks) on existing temporary files allocated to mass storage device, more memory, allocation of new mass storage files, or growth of existing permanent files. It may also be used to create a new file or to access an existing permanent file. The user must be prepared to accept a denial return if the request cannot be satisfied.

Entry to this routine is made from the fault vector as a result of a MME GEMORE. At the time of the execution of this MME, the Q-register must contain the pertinent file code in bits 24-35, unless the request is for memory.

The calling sequence for MME GEMORE is as follows:

```
L      MME      GEMORE
L+1    ZERO     A,B
L+2    Denial return
L+3    Successful Return
```

MME GEMORE may not be used in courtesy call. The user must be prepared for denial.

MME GEMORE For Additional Memory

To request additional memory space, the L+1 in the MME GEMORE calling sequence is as follows:

A - 0
B - Number of 1024-word blocks requested

If the request for additional memory is satisfied, the additional blocks of memory are added to the high address end of allocated memory. It is suggested that the maximum memory requirements for a given activity be requested initially (prior to allocation). Up to 64K words of memory can be requested on one MME GEMORE request, however large requests may cause a job to be held out of execution until the total space is available.

A MME GEMORE for memory may be denied for the following reasons:

1. Request for more than 64K of memory.
2. Requested memory size would be so large that job could not be loaded between Program 01 and End-of-Quadrant 0.
3. Not enough mass storage available to hold Swap file.
4. Outstanding abort request for the program.

If sufficient space is available contiguous to the requesting job, the requested amount of space is added to the program. If sufficient contiguous space is not available, the program is swapped out of memory, and the requested space is added when the job returns to memory.

MME GEMORE For Additional Magnetic Tapes

To request an additional magnetic tape handler:

C(Q)	0-17	Zero, or pointer to a word containing the reel number of the tape
	24-35	Contains file code (FC) to be used

L+1 is as follows:

A = 1 for 7-track tape
  = 3 for 9-track tape

B = Scratch/Input designator:

0 - Only a tape unit in READY status is acceptable.
1 - A unit in READY or STANDBY status is desired. A MNT instruction output on the console directs the operator to mount a reel. The reel number is contained in the location indicated by 0-17 of Q. If Q(0-17)=0, reel #99999 is used.

Default disposition codes are as follows:

R - For READY tapes
D - For STANDBY tapes

NOTE: The density field refers to the capability of the device, not its physical density setting. Programs which use this MME are responsible for setting the density. This permits the GFRC and UFAS "OPEN" procedures to read a pre-existing label on the tape, if one exists, even if it is written at a density other than that specified. GFRC and UFAS reset the density at the conclusion of their "OPEN" processing to that requested in the original MME.

To request an additional tape unit (MTS500, MTU0400, MTU0500, or MTU0600):

```
C(Q)    0-17  Zero, or pointer to a word containing the reel number of
              the tape
       18-19  Unused
       20-23  Density codes as follows:
                0000 - site standard high density
                0001 - 200 bpi ┐
                0010 - 556 bpi ├   NRZI Mode
                0100 - 800 bpi ┘
                1001 -1600 bpi     PE Mode
                1100 -6250 bpi     GCR mode
                1111 - Tape handler capable of all densities
                       for requested seven or nine track tape
                0111 - Any tape of any density available
       24-35  Contains File Code (FC) to be used
```

L+1 is as follows:

A = 1 for 7-track tape
  = 3 for 9-track tape

B = Scratch/Input

0 - Only a tape unit in READY status is acceptable.

1 - A tape unit in READY or STANDBY status is desired. A MNT instruction output on the console directs the operator to mount a reel. The reel number is contained in the location indicated by bits 0-17 of the Q-register. If bits 0-17 of the Q-register = 0, reel #99999 is used.

Default disposition codes are as follows:

R - For READY tapes
D - For STANDBY tapes

MME GEMORE For Additional File Space

To request additional links of file space for an existing temporary or permanent file, the contents of the Q-register must be defined as follows:

Bits	0-23	Zero
	24-35	File code of existing file

The content of L+1 in the MME calling sequence must be as follows:

A - 2

B - Number of links (link = 3840 words) to be added to the specified file. For a permanent file, if B = 0, the File Management Supervisor determines the amount of space the specified file is to be given.

A successful return is made when an additional link can be allocated for a sequential (linked) file. For a random file, a successful return is made only if the requested number of links has been allocated.

REQUESTING NEW TEMPORARY FILE

To request the allocation of a new temporary file via MME GEMORE, the content of the Q-register must be as follows:

Bits	0-16	Zero
	17	=1 for random file =0 for sequential file
	18-23	Zero
	24-35	File code of file to be created

The content of L+1 in the MME calling sequence must be as follows:

A - 2

B - Number of links for the new file

Example:

```
        LDQ     =3HOFC,DL       File code
        ORQ     =1,DU           Random indicator
L       MME     GEMORE
L+1     ZERO    2,10            Request 10 links
L+2     TRA     None            Denial return
L+3                             Normal return
```

A user can also use a MME GEMORE for a null file to be used as an EOF stream and output sink. The MME GEMORE is for a temporary file of zero links.

For example:

```
        LDQ     =3HOFC,DL       (file code)
L       MME     GEMORE
L+1     ZERO    2,0             (mass storage file, zero links)
L+2     TRA     none            (denial return)
L+3     Normal exit             (normal return)
```

The PAT and PAT pointer built for the null file will be the same as for a $ FILE fc,NULL card.

REQUESTING CATALOGED FILE

To request a cataloged file via MME GEMORE, the content of the Q-register must be defined as follows:

Bits 0-17 - Pointer to qualified filename block
18-23 - Access permissions as follows:

18	19	20	21	22	23	Requested Permission
1	0	0	0	0	0	Read
1	0	0	0	0	1	Read/C (C = Changing)
0	1	0	0	0	0	Write
0	1	0	0	0	1	Write/C
1	1	0	0	0	0	Read/Write
1	1	0	0	0	1	Read/Write/C
1	1	1	0	0	0	Load (L)
1	1	1	1	0	0	Recovery
0	0	0	0	0	1	Query
0	0	0	0	1	0	Test
0	0	0	0	1	1	Test/C
0	1	0	1	0	0	Private

25-35 - File code

NOTE: MME GEMORE cannot be used to request cataloged, nonstructured media files.

The permissions are summarized as follows (for complete details of file permissions refer to the File Management Supervisor reference manual):

Read or R	- Read only from a file while it is not being changed.
Read/C or R/C	- Read from a file while it is being controlled to limit the effect of the change on the reader.
Write or W	- Read from and write to a file while it is not being changed, and deny allocation to any readers requiring control of the writer, to limit writer affecting readers.
Write/C or W/C	- Read from and write to a file while it is being controlled to limit the effect of other writers on this writer, and the effect of this writer on other readers and writers.
Read/Write or R/W	- Same as Write.
Read/Write/C or R/W/C	- Same as Write/C.
Load or L	- Allows allocation of a protected file without invoking collection of "befores" and/or "afters", e.g., in the initial loading of a file or during recovery when there is no need to collect before and/or after images. The requested file is allocated (if not in use) for read/write. Any ABORT/ROLLBACK/, RDERR/JOURNAL/, or ACCESS/MONITOR/ options are ignored. If the file has none of these options, allocation is treated as a request for Write (W) allocation. The RDERR/DUP/ and VERIFY/YES/ options are honored, if present.

Recovery	- Read from and write to a file even when the file is locked to other allocations because of bad file content and while file is not being recovered by another user.
Query	- Read from a file while it is being changed whether or not it is controlled to limit the effect of writers on the reader, and whether or not the file is locked because of bad file content.
Test or T	- Read from a file while it is not being changed and write to an auxiliary file from which changed parts are read.
Test/C	- Read from a file while it is being controlled to limit the effect of changes on readers, and write to an auxiliary file from which changed parts are read.
Private or P	- Provides exclusive Write access by preventing other users from reading from a concurrent access file while the file is being written.

L+1 in the calling sequence is as follows:

A - 4 - Request cataloged file. Access mode is same as that in which file was created.

5 - Request cataloged file. Access mode is random.

6 - Request cataloged file. Access mode is linked.

B - Pointer to 355-word buffer

If the request is denied, bits 1-11 of the Q-register contain the reason code.

The FMS return message is available in word 43 (octal) of the 355-word user buffer. Word 37 (octal) contains the location of the message in bits 0-17 and the length of the message in bits 18-35. Refer to Appendix F in the File Management Supervisor Reference Manual for a list of the return codes and messages.

A $ USERID card must be in the job deck prior to the activity requesting cataloged file.

The qualified filename block contains a four-word entry for each level in the catalog structure as follows:

$\left.\begin{array}{l}0\\1\end{array}\right\}$ file/catalog name i

$\left.\begin{array}{l}2\\3\end{array}\right\}$ associated password

If the first word (user master catalog name) contains a minus one (octal 777777777777), FMS supplies the user master catalog name from .SUID. This is compatible with the DRL FILACT capability of the Time Sharing System and enables a user to develop more generalized routines for permanent file management.

Each name and each password must be left-justified, with trailing blanks if less than 12 characters. In addition, the last entry of the block must be followed by a word of all 1 bits:

```
          0}    file name
          1}
          2}    password
          3}
      fence     777777777777
```

The 355-word buffer is necessary to hold the catalog descriptors during FMS processing of the request. This buffer is masked before return to the caller, in order to maintain file system security. However, if the request is denied, the FMS error message is left in the buffer. (This allows programs to use standard error messages.) A roadblock is issued before any FMS access to prevent user courtesy calls from sampling this buffer during I/O activity.

During the time the file system modules are active, and until the 355-word buffer area can be cleared, the dump option is masked to prevent disclosure of catalog blocks in case of activity abortion.

REQUESTING FILE CONCATENATION

To request file concatenation, the Q register must be defined as follows:

Bits	
0-17	address of a 40-word work area
18-23	zero
24-35	file code

The content of L+1 in the MME calling sequence must be as follows:

A - 7
B - address of a 355-word buffer

Three exits may be taken. L+2 is taken if the current file is the last file *(true EOF). L+3 is taken when a PAT entry describing the file exists and the file code has been altered to the users file code. Normal file operations can continue without additional system support. L+4 is taken when the next file has not been preallocated. When L+4 is taken, the first two words of the user's work area (address in bits 0-17 of the Q register) contains the parameters to request file allocation (MME GEMORE code 4):

```
Word

 1       Zero     4,B
 2       VFD      18/C,06/40,H12/fc
```

where:

b = the 355-word buffer address
c = pointer to catalog string
    developed in the work area

In word 2, read permissions are requested in bits 18-23 and the file code is contained in bits 24-35.

GEMREL -- RELEASE MEMORY

MME GEMREL is used to deallocate a specified amount of memory from a requesting program's total assigned memory. Actual memory deallocation is in multiples of 1024-word blocks.

Entry to this routine is made from the fault vector as the result of a MME GEMREL. The A- and Q-registers must contain entry parameters.

The calling sequence is as follows:

```
L      MME     GEMREL
L+1    Error Return
```

Return is to the address specified in the A-register. When lower memory is released, there is a change in the base address of the requesting program. In this case, the return address is assumed to be relative to the base address after memory deallocation. When upper memory is released, the base address of the requesting program is not changed.

The number of words released is truncated to a multiple of 1024. The contents of word 37 (octal) of the slave program prefix is not changed by this MME.

Initially: C(A) required:

Bits	0-17	Return address
	18-35	Ignored

C(Q) required:

Bits	0-17	Number of words of lower memory
	18-35	Number of words of upper memory

Release of lower memory should only be attempted by load type routines because of base address relocation. MME GEMREL may not be used during courtesy call.

GENEWS -- SPAWN NEW JOB

MME GENEWS allows a user program to spawn programs for execution by the system. The program(s) to be spawned may be described on a temporary file or within the originating program's allocated main memory.

Entry to this routine is made from the fault vector as a result of MME GENEWS.

The calling sequence is as follows:

```
L        MME      GENEWS
L+1      ZERO     A,B
L+2      Abort Return
L+3      Normal Return
```

Where: A - Pointer to the first word of a 320-word buffer for use by the MME GENEWS processor.

B - Pointer to a location containing either:

1. File code in bits 24-35. If the job to be spawned resides on a temporary file, bits 0-17 must be zero.

2. Starting location of an input stream in bits 0-17 if the job to be spawned resides in memory.

On a normal return, the Q-register contains the left-justified, zero-filled SNUMB of the job which was spawned.

Spawning A Job Residing On A Temporary File

The temporary job may be loaded with the job to be spawned as follows:

```
$  SNUMB     NNNNN
$  IDENT     XXXXX,....
     .                           )
     .                           } Job containing MME GENEWS
     .                           )
$  DATA      FC,,COPY
$  DUMMY     GENEW
$  IDENT     XXXXX,...
$  OBJECT
     .         )
     .         } Job to be put on spawn file
     .         )
$  DKEND
$  ENDJOB
$  ENDCOPY
```

All cards between $ DATA and $ ENDCOPY are loaded on file FC.

The job on file FC may be spawned as follows:

```
   MME      GENEWS
   ZERO     A,B
   TRA      ERROR
    .
    .
B  VFD      24/0,H12/FC
A  BSS      320
```

Spawning A Job Residing In Memory

The job skeleton is described within memory as a BCI string. When spawning jobs residing on permanent files, the BCI input stream normally terminates with a $ SELECT. For job skeletons residing totally within memory, the stream terminates with the $ ENDJOB.

To spawn the job on permanent file AB, with catalog name JDOE and password OPEN, the coding could be as follows:

```
     MME      GENEWS
     ZERO     A,B
     TRA      ERROR
      .
      .
  C  BCI      3,$ƀƀƀƀƀƀIDENTƀƀƀXXX
     BCI      2,XX,JOHNƀDOE
     BCI      9,
     BCI      3,$ƀƀƀƀƀƀSELECTƀƀJDO
     BCI      5,E/AB
     BCI      6,
     ZERO     -1,0
  A  BSS      320
  B  ZERO     C,0
```

MME GENEWS Parameters

The following parameters apply to this MME:

1. On temporary file jobs, the $ DUMMY control card must be the first card image.
2. On in-memory jobs, the BCI string must start with either the $ IDENT control statement image, or with the $ SELECT control statement image which contains the IDENT.
3. All BCI card images must be 14 words long. The GMAP restriction of nine words for the BCI pseudo-operation must be observed.
4. The end of the BCI input stream is denoted by: ZERO -1,0.
5. If the file code is zero or if either character of the file code is an * or a $, the request will be aborted.
6. The $ SNUMB and activity number of the originating job appear on the $ SNUMB control card of the spawned job starting in column 36.
7. The $ DUMMY card image for temporary file jobs is replaced by a valid $ SNUMB card image of the job to be spawned. The SNUMB of the spawned job will be in the Q-register on a normal exit. In jobs utilizing the BCI input stream, the $ IDENT image is prefaced by a valid $ SNUMB image.
8. There is no restriction on the types of jobs to be spawned. Any job which runs normally can be spawned.
9. A job spawned from a remote station is returned to that station.
10. In an abort return, the first word of the buffer contains an abort code (right-justified, binary) indicating the abort reason.

CODE	REASON
1	Not applicable.
2	File/stream pointer outside calling program.
3	Illegal file code for temporary file.
4	File code not in PAT.
5	Temporary file device not mass storage.
6	Nonrecoverable I/O error.
7	No mass storage space available.
10	First temporary file block not $ DUMMY card image.
11	No room for PAT body.
12	System Scheduler queue full.
13	Not applicable.
14	End-of-Stream flag out of position or missing.

GEPRIO -- I/O PRIORITY

MME GEPRIO is used to assign I/O priority for a magnetic tape file. The file code in the calling sequence is assigned priority over the user's other magnetic tape files in the I/O queue.

Entry to this routine is made through the fault vector as a result of a MME GEPRIO in a user program.

The calling sequence for this routine is as follows:

```
L      MME     GEPRIO
L+1    Return
```

```
Initially:  C(Q) required

          Bits  0-23   Not used
                24-35  File code
```

The MME GEPRIO applies only to magnetic tape and can only be used for one file code at a time.

GERELC -- RELINQUISH CONTROL

The MME GERELC entry causes the program to be taken out of execution until any one of its I/O requests has been serviced. If there are no outstanding I/O requests, the program is returned to the Dispatcher queue.

Entry to this routine is made from the fault vector as the result of a MME GERELC.

The calling sequence for this routine is as follows:

```
     L      MME     GERELC
     L+1    Return
```

GERELS -- COMPONENT RELEASE

MME GERELS is used to deallocate peripherals, to alter disposition codes, and to release a portion of a temporary mass storage file from a program. For each file code listed in the calling sequence, the corresponding peripheral file is deallocated from the requesting program. MME GERELS can be used to alter the disposition codes for normal activity termination and/or activity abort termination. The file disposition codes are one or two alphabetic characters appended to the Logical Unit Designator (LUD). (The LUD is a two or three character alphanumeric device name that must be used if files are to be saved or dismounted).

Either one or two disposition codes can be used. The first disposition specifies component disposition on normal termination. The second disposition code specifies component disposition on an abort termination. Thus, disposition codes SR specify that on normal termination the file is to be saved for use in a subsequent activity (S disposition) and is to be released if the activity aborts (R disposition). Allowable disposition codes are as follows:

1. For all file storage devices-

   R - Release file (implied if not specified)
   S - Save file for subsequent activity
   P - Purge file contents and release file

2. For magnetic tape and disk pack files-

   D - Dismount medium from system (for disk packs, this is automatically converted to R disposition)

   C - Write inhibit the medium and save for subsequent activity.

Entry to this routine is made from the fault vector as the result of the MME GERELS. The calling sequence to deallocate peripherals is as follows:

```
     L        MME   GERELS
     L+1      ZERO  Number of files (N) in upper half of word
                    Alter flag in lower half of word
                    Flag = 0, deallocate peripherals
     L+2      BCI   1,0000F1
     L+3      BCI   1,0000F2
     .
     .              F1, F2,...., FN-file codes
     .
     L+N+1    BCI   1,0000FN
     L+N+2    Return
```

If this file is on magnetic tape and has a normal disposition code of C, a Set File Protect (SFP) command is issued, or a file protect message is output for this file. If the file is on magnetic tape and has a disposition code of R, the tape is rewound using the Rewind instruction, dismounting instructions are not issued, and the tape is released to GCOS for reassignment. If the file is on magnetic tape and has a disposition code of D, the tape is rewound using Rewind and Standby, the operator is issued dismount instructions, and the tape unit is released to GCOS for reassignment.

If this file is a nonstructured disk pack, a Dismount message is issued to the console and the unit is designated not assignable. When the pack has been removed, an operator ASGN request is required to return the spindle to service.

For all other devices, files with disposition codes of D or R are treated as R. If C is specified for nontape devices, it is treated as S. No action is taken if the file has an S disposition code; the file is not rewound or released, and no indication is given to the user that the request was ignored. If more than one file is assigned to a device (or storage space), the resource is deallocated only when the last remaining file is released. The content of L+1 in the MME calling sequence identifies the number of file codes appearing in the calling sequence.

The calling sequence to alter disposition code is as follows:

```
L       MME     GERELS
L+1     ZERO    N,AF
L+2     BCI     1,I00cfx
L+3     BCI     1,I00cfx
.
.
.
L+N+2   Return

Where:    N - Number of files (in upper half of word).
         AF - Alter flag (in lower half of word).  If flag ≠ 0,
              the disposition code is altered.
          I - = 0 Alter normal disposition code
              = c Alter abort disposition code and c is one of the
                  following substitute disposition codes:
                     R - Release
                     S - Save
                     C - Continue
                     D - Dismount
                     P - Purge
                     L - Leave
                     M - Mode
         fx - File codes
```

Either the normal or abort disposition code for the file FN can be altered at any convenient time in the program including during wrapup (for changing abort disposition codes).

When a file disposition is modified to Save or Continue, a check for an existing Logical Unit Designator (LUD) is made. If none is found, the file code becomes the LUD. Thus, a file may be created by MME GEMORE, set to S disposition by MME GERELS and accessed in a subsequent activity by using the original file code as the LUD. In this case, the second character of the original file code must be numeric or an *: it cannot be alphabetic.

The calling sequence to alter access mode is as follows:

```
L       MME     GERELS
L+1     ZERO    1,1
L+2     BCI     1,00RMF1
```

When the character preceding M equals R, the access mode is random. When it is any other character (e.g., L) the access mode is linked.

The calling sequence to purge implicit files is as follows:

```
L       MME     GERELS
L+1     ZERO    1,1
L+2     BCI     1,000PF1
```

(Used when PURGE was omitted from the activity control card.)

```
L       MME     GERELS
L+1     ZERO    1,1
L+2     BCI     1,0000F1
```

(Used when PURGE is on the activity card.)

```
L       MME     GERELS
L+1     ZERO    1,1
L+2     BCI     1,000LF1
```

(Used when PURGE is on the activity card but file is not to be purged.)

A disposition of P causes the file to be purged at release time. L disables any purge on the file and leaves the data as-is at release.

Use of P in either the normal or abort disposition field causes the file to be purged at release time even if that release may occur in a subsequent activity of the job. Files are purged only if they have Write permission and, in the case of cataloged files, the user must be the creator of the file. Files with no I/O activity during the job are not purged. When possible, hardware write-protected files should be dismounted instead of purged.

The calling sequence to release temporary mass storage file is as follows:

```
L       MME     GERELS
L+1     ZERO    1
L+2     VFD     18/n,6/0,H12/FC
L+3     Return
```

Where: $n = 0$, Total file is released
$n \neq 0$, Partial release of the last n links of the file
If $n \geq$ total links of the file, all but one link is released.

The calling sequence to release cataloged files is as follows:

```
     Initially:         C(Q) = (Ptr (b)),-1

          L         MME        GERELS
          L+1       ZERO       n,0
          L+2       BCI        1,0000F1
          L+3       BCI        1,0000F2
           .
           .
           .
          L+n       BCI        1,0000Fn
          L+n+1     Return

          Where:    b = location of 355-word buffer area
                    F1, F2, Fn are file codes
```

Cataloged and noncataloged files may be intermixed in the calling sequence if desired.

In the release sequence, .MRELS determines that the file is cataloged and calls the Perm File Drop module. The 355-word buffer pointer is validated and the activity aborted if it lies outside slave limits or below 100 octal. If the pointer is valid, a queue is obtained and a type 6 call is made to the De-Access File System module. As in the case of attachment, a roadblock is used and the Dump bit is masked during FMS activity. (For return codes and messages, refer to Appendix F in the <u>File Management Supervisor</u> Reference Manual. The FMS return message may be located by a DCW in word 37 (octal) of the 355-word user buffer. Bits 0-17 contain the location and bits 18-35 contain the length.)

If a cataloged file is specified, which has ABORT/ROLLBACK/protection and a page size greater than 320 words, the buffer provided must be "35 + page size" words instead of 355 words.

MME GERELS may not be used during courtesy call.

When a file with an S disposition is the object of a MME GERELS in the last activity of a job, that file is saved until the end of that activity and then released.

<u>GERETS -- RESET SWITCH REQUEST</u>

MME GERETS is used to reset bits in the program switch word according to corresponding bits in the Q-register. For each bit position of the Q-register that contains a 1, the corresponding bit of the switch word is set to 0. For each bit position of the Q-register that contains a zero, the corresponding bit of the switch word is not changed. The resultant setting of the switch word is returned in the Q-register.

Entry to this routine is made from the fault vector as the result of a MME GERETS. The standard MME call input parameters pertain.

The calling sequence is as follows:

```
L      MME     GERETS
L+1    Return

Initially:    C(Q) = 36-bit mask

Results:  C(Q) = Contents of switch word
```

<u>GEROAD</u> -- <u>ROADBLOCK</u>

MME GEROAD causes the requesting program to be taken out of execution until all of its outstanding I/O requests (including courtesy calls) are completed. If there are no outstanding requests, the program is placed in the Dispatcher queue.

Entry to this routine is made from the fault vector as the result of a MME GEROAD. The calling sequence is as follows:

```
L      MME    GEROAD
L+1    Return
```

MME GEROAD may not be used during a courtesy call.

<u>GEROLL</u> -- <u>RE-INITIATE OR ROLLBACK PROGRAM</u>

MME GEROLL is used to restart a program from a previously established checkpoint. The use of MME GECHEK and MME GEROLL are described in detail in the <u>Program Recovery/Restart</u> manual.

The calling sequence for this routine is as follows:

```
MME   GEROLL
```

After the rollback is completed, control is returned to the address specified in location 14 (decimal) of the slave program prefix in the user's assigned memory. If at the time a checkpoint is taken (via MME GECHEK), bits 0-17 of this location are zero, the checkpoint processing places the address of the MME GECHEK plus one into this location.

When a MME GEROLL is executed, the checkpoint file content is read. This includes an image of the slave program and parts of its SSA area at the time of the checkpoint and also data file positioning information. The information from the file is placed into the slave program area so that the program can be restarted from its status at the time of the checkpoint execution.

Rollback always uses the positioning information from the last checkpoint dump. Positioning to selected dumps is not possible.

The contents of the machine registers are restored to the point immediately prior to the last checkpoint dump. Positioning of magnetic tape files and data files is accomplished using information saved at checkpoint time. The File Management Supervisor is called by the MME GEROLL processor to restore protected files to their proper state.

All user data files that were present when the checkpoint was written must be present at rollback time. The only exception to this is when the MME GERELS list was provided in the MME GECHEK (see MME GECHEK additional calling sequence options).

During rollback, the amount of memory currently allocated to the slave program must equal the amount of memory allocated at the time of the checkpoint. If these are not equal, rollback will get up to 64K more for the job. If more than 64K is needed, the slave program is aborted.

Repositioning of multireel tape files where the wrong reel is mounted at rollback time will be done for programs using File and Record Control or Unified File Access System (UFAS). If the wrong reel is mounted at rollback time, the operator will be given the required reel sequence number so that the proper tape can be mounted.

The MME GEROLL must not be used in courtesy call.

MME GEROLL For Deferred Restart

MME GEROLL may be used in conjunction with MME GECHEK to perform job suspension with deferred restart. MME GECHEK is used to place the job on a specified permanent file. When a MME GEROLL instruction is encountered in a subsequent job, the suspended job (the job placed on mass storage by the MME GECHEK) is restarted from the checkpoint taken in the previous run. The checkpoint file must be a permanent file in order to be used for a deferred restart.

The user's program deck must contain all file, peripheral device, and memory requirement definitions. The program must also be coded to perform the following:

1. Store the following information in bits 0-17 of word 15 of the slave program prefix.

   For programs using File and Record Control -

   ```
   Bit(s) 0-17 - Checkpoint file control block location
   ```

   For programs not using File and Record Control -

   ```
   Bit(s) 0-5  - Not used
          6-17 - File code of checkpoint file
   ```

   The following flags must be stored in the lower half of location 14 of the slave program prefix:

   ```
   Bit 19 = 0 File control block pointer in location 15
          = 1 File code in location 15
       20 = 1 Deferred restart.
   ```

2. Execute the MME GEROLL instruction.

For example, the following JCL might be used to restart a job:

```
1      8          16
______________________________
       .
       .
       .
$      ENTRY      START
$      GMAP
       SYMDEF     START
START  LDA        =0300000,DL
       STA        14
       LDA        =3HO(file code),DU        Checkpoint file code
       STA        15
       MME        GEROLL
       END
$      EXECUTE
$      LIMITS     ....
$      PRMFL      file code,....            Checkpoint file code
       .
       .
       .
$      TAPE
       .
       .
       .
$      DATA       I*                        Data storage file
       .
       .
       .
       Data continuation
       .
       .
$      ENDJOB
```

NOTE: Even though the job uses File and Record Control, the job may be restarted using the file code designation for the checkpoint file.

The file and peripheral device definitions in the original job and in the subsequent job (the job containing the MME GEROLL instruction) must be compatible. The $ LIMITS card in the subsequent job should designate the maximum memory size needed for the job.

All activities subsequent to the checkpoint activity in the initial job must be included after the rollback activity in the deferred restart job. The J* and *J files are not saved by the MME GECHEK and must be rebuilt by System Input for the unprocessed activities.

If SYSOUT files are not closed (flushed) before the MME GECHEK is executed, the deferred restart will contain the last buffer of File and Record Control output, and some output may be duplicated. If the SYSOUT files are closed before the checkpoint, they must be opened after the rollback operation.

Caution must be exercised in initiating the deferred restart. The user must consider the possibility that the files containing the original job may have been changed since the initial execution of the job. That is, files that can be accessed by more than one job may have changed so that the positioning information for the restart is no longer valid.

Table 4-1. MME GECHEK/GEROLL Characteristics

Characteristic	MME GECHEK/ MME GEROLL	Deferred Restart
Checkpoint File -		
Temporary (mass storage)	Yes	No
Permanent (mass storage)	Yes	Yes
Tape	Yes	No
Designation - CHKPT file -		
File code	Yes	Yes
File control block (open)	Yes	Yes (see 1 below)
Position		
SYSOUT	Yes	N/A
Tape files	Yes (see 2 below)	Yes (see 2 below)
Multireel files	* (see 3 below)	* (see 4 below)
Permanent files (linked)	Yes	Yes
Protected files	Yes	Yes
Temporary files	Yes	Yes (see 5 below)
System temporary files	Yes	No
GEMORE - Allocate -		
Permanent files	* (see 3 below)	No
GEMORE - Create -		
Temporary files	* (see 3 below)	No
GERELS after Startup	* (see 3 below)	* (see 6 below)

---

1. File control block must exist in the program executing MME GEROLL.
2. Maximum 21 tapes.
3. MME GECHEK should be taken after MME GEMORE/GERELS or reel exchange.
4. User must be responsible for proper reel mounted before MME GEROLL.
5. User must reconstruct contents if necessary for continued execution.
6. User defines all peripheral devices and files in restart deck.

MME GEROLL Error Codes

An error during the MME GEROLL execution aborts the program with the lower half of the Q-register containing an error code (in octal).

Code	Error Description
1	File or checkpoint not found
2	Incompatible memory size and sufficient amount could not be obtained
3	No checkpoint taken
5	Bad read status on checkpoint file
7	Invalid checkpoint file device
10	MME GEROLL cannot be executed because protected file cannot be repositioned
11	Undefined file for deferred restart
12	Cannot perform MME GEROLL in courtesy call
13	File too fragmented to reposition
14	Number of SSA's has changed

NOTE: MME GECHEK error codes are described under GECHEK -- Checkpoint Dump.

If the rollback operation is successful but the lower half of the Q-register contains a 4, a tape(s) is no longer assigned or positioning was lost. The user must decide whether to abort on this condition.

GEROUT -- REMOTE OUTPUT RECORD

A user program executing in the direct program access (DAC) mode uses the MME GEROUT instruction for the following types of functions:

- o Send output to and receive input from a terminal
- o Identify a terminal by station identification
- o Request a line status or a line disconnect
- o Switch operating modes

Each MME GEROUT function is identified by a unique operation code that must be specified in the MME GEROUT calling sequence. Entry to a MME GEROUT is made via the fault vector as the result of the execution of the MME GEROUT instruction.

NOTE: The MME GEROUT instruction is used to access the General Remote Terminal Supervisor (GRTS) and the Network Processing Supervisor (NPS). In the descriptions of the MME GEROUT functions, the generic term communications subsystem is used for these programs.

The general format is as follows (exception is the terminal type):

```
L      MME    GEROUT
L+1    VFD    18/Record Pointer,06/OP,H12/Terminal Identification
L+2    ZERO   Status Word Pointer,Courtesy Call
L+3    Next Instruction
```

Where: L+1 is defined as follows:

Bits	0-17	Record pointer
	18-23	Operation code
	24-35	Terminal identification

In issuing MME GEROUT instructions, the same code must not be used for both main level and courtesy call level. This is necessary because the MME GEROUT processor may be uninhibited and can be interrupted. The interrupt can be for I/O for this program and can be due to a previous MME GEROUT with a courtesy call specified in the calling sequence. In this case, the MME GEROUT calling sequence in the main level code is changed. When control is returned to the interrupt point in the main level code, the MME GEROUT calling sequence will not contain the expected data for continuation of the program. Instead, it will contain data from the last MME GEROUT issued (the one issued in courtesy call).

Refer to Remote Terminal Supervisor (GRTS) reference manual or Network Processing Supervisor (NPS) reference manual for the format of the record pointed to by bits 0-17 of L+1.

Direct Access Output

The Direct Access Output MME, operation code 03, allows a user program in execution to send output to one or more terminals. The calling sequence is as follows:

```
L      MME    GEROUT
L+1    VFD    18/Record Pointer,06/03,H12/Terminal Identification
L+2    ZERO   Status Word Pointer,Courtesy Call
L+3    Next Instruction
```

The record pointer designates the symbolic location of word 0 of the data block, which contains the number of words in the block and the media code of output destined for a voice-grade terminal. If transmission is to a G-115 terminal, the format of words 1 to n is one print line, using six-bit characters, with an octal 77 indicating the end of print line and the next character used as the slew character. If transmission is to a teletypewriter or keyboard-display terminal, word 0 does not include a media code, and word 1 contains the count of the characters to be transmitted. Words 2 to n contain the actual characters to be transmitted. Each character is right justified within a nine-bit field. Characters in a partial word are left-justified in the word.

A two-character code in bits 24-35 of L+1 identifies the terminal.

The status word pointer points to a word that contains the status of the operation in process. The format of the word is as follows:

Bits 0-17	Not used
18-23	VIP Terminal hardware status: 18-21 = 0 22 = 1 PRINT key pressed to initiate transmission. 23 = 1 Transmission failed after 16 attempts.
24-26	Not used
27	Abort (in general status mode, Break/Disconnect waiting)
28	Not used
29	Connected to another slave program
30	Waiting to connect to slave program
31	Terminal idle
32	If = 1, communication modules in the central computer system cannot acknowledge I/O request because terminal is busy with I/O.
33	If = 1, output transmission completed.
34	If = 1, terminal has not called in and connected or has disconnected
35	If = 1, operator sent BREAK message which terminated operation.

The courtesy call address is used.

Direct Access Output, Then Input

This MME GEROUT function allows the user program to transmit a data message to a terminal and to receive a data response from that terminal. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/Record Pointer,06/04,H12/Terminal Identification
L+2    ZERO    Status Word Pointer, Courtesy Call
L+3    Next Instruction
```

In addition to the requirements for the DAC Output MME GEROUT, the user program must provide an input buffer address in the lower part of word 0 of the data block for this MME instruction. An input buffer of at least 31 words is required for input from a teletypewriter terminal. This is necessary to accommodate paper tape input. An input buffer of 520 words is required for input to accommodate all VIP devices. If a smaller input buffer size is desired, MME GEROUT (44) must be used to notify the FNP.

The record pointer designates the symbolic location of word 0 of the data block, which contains the number of words in the data block, the media code of the output, and the address of the input buffer.

Operation Code 04 specifies output, then input during direct access with a user program.

The status word pointer points to a word that contains the status of the operation in progress. The status format is the same as for Direct Access Output, except that bit 33 = 1 indicates output transmission is completed and the input received has been placed in the user's input buffer.

User Program Inquiry To Terminal

When a user program is ready to establish communication with any terminal that requests direct access with it, the user program enters a program identification name in L+2. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/0,06/05,H12/Terminal Identification
L+2    BCI     User program Identification
L+3    Next Instruction
```

The communications subsystem modules in the central computer system retain a list of all terminals that have transmitted a Direct Program Access (DAC) control record. Thus, when a program requests connection to a remote terminal, this list is scanned for that terminal's identification. When a match of names is found, the terminal identification is placed in the right 12 bits of the word directly after the MME GEROUT. The pointer to this MME is then deleted from the list, and the user program can send an appropriate message directly to the terminal.

If a match in identification cannot be found, the request is retained until a terminal calls in and specifies the user program identification. If a terminal calls in at a later time and wishes to connect to that user program, the communications subsystem establishes the connection and inserts the station identification in the user program's outstanding MME. Thus, the remote inquiry request is active until a positive response can be given to the user program. Control is returned to the user program whether or not a new terminal has been connected.

The user program may have two outstanding remote inquiries at a time. The inquiries must be issued from different locations and have different inquiry names. Any other inquiry results in the program being aborted.

To determine whether a positive response was received, the user program should initialize (to zero) the terminal identification field of the MME calling sequence prior to using the remote inquiry.

A remote terminal, which does not specify a user program identification when it issues a DAC command, is connected only to a slave program which issues a MME GEROUT specifying the terminal by its station identification. Terminal connection in this case is made by station. MME GEROUT instructions which can use this connection are:

DAC Output
DAC Output/Input
DAC Output/Prepare PPT

Operation code 05 specifies user program inquiry regarding direct access.

The BCD user program identification is a six-character BCD word by which the user identifies his DAC program.

Program Requests Terminal Type

When connected to a terminal, the user program can determine the terminal type by using the Program Requests Terminal Type MME GEROUT function.

```
        L       MME       GEROUT
        L+1     VFD       6/Type,12/0,06/06,H12/Terminal Identification
        L+2     Terminal Not Connected
        L+3     Next Instruction
```

Type indicates whether it is a remote line printer, document handler, remote computer terminal, a teletypewriter, or keyboard-display terminal.

Operation code 06 specifies a user program request for the terminal type. The communications subsystem stores the terminal type in bit positions 0-5 of L+1, when it encounters a 06 operation code.

The device type codes (in octal) are as follows:

DATANET 355/6600 Front-End Network Processor -

01	Reserved for the system
02	Reserved for the system
03	Remote computer terminal
04	Teleprinter
05	DATANET 760 VIP - Screen size = 4 lines x 46 characters
06	DATANET 760 VIP - Screen size = 8 lines x 46 characters
07	DATANET 760 VIP - Screen size = 16 lines x 46 characters
10	DATANET 760 VIP - Screen size = 26 lines x 46 characters
11	VIP Series 765/775 - Screen size = 22 lines x 46 characters
12	VIP Series 785 - Screen size = 22 lines x 92 characters
13	VIP Series 7700 - Screen size = 12 lines x 80 characters
14	VIP Series 7700 - Screen size = 22 lines x 46 characters
15	VIP Series 7700 - Screen size = 24 lines x 80 characters
16-17	Reserved for the system
20	2741 Teletypewriter
21-27	Reserved for the system
30	MRS200 Document Handler
31	DRD200 Document Handler
32	DRD236 or DHU1604/8/12/16 Document Handler
33-47	Reserved for the system
50-60	Reserved for T and D
61-77	Reserved for customer use

The primary use for this MME GEROUT is to determine if the terminal is still connected to the user program. If the user has issued a DAC I/O request for this terminal, the associated I/O status word indicates whether or not the terminal has disconnected.

The return from the terminal type MME request is to location MME+2 if the terminal has disconnected, and to MME+3 if the terminal is still connected to the user program.

Program Requests Line Disconnect

A user program can request that the communications subsystem disconnect a particular remote terminal. The calling sequence is as follows:

```
L       MME     GEROUT
L+1     VFD     18/0,06/17,H12/Terminal Identification
L+2     ZERO    Status word pointer,Courtesy Call
L+3     Next Instruction
```

Operation code 17 (octal) specifies a line disconnect request.

Terminal identification is a two-character code that identifies the terminal to be disconnected.

Status word pointer points to a word that contains the status of the operation in process. This word format is all zero if a disconnect is in progress, or bit 34 = 1 if a disconnect is completed.

In response to this MME, the communications subsystem sends a request to the DATANET processor to disconnect the station specified by the user program. The user program must not issue further I/O commands to the station while the disconnect MME is in progress.

Courtesy Call is used.

Direct Access Current Line Status

A user program can request the current line status of a particular remote terminal.

```
L       MME    GEROUT
L+1     VFD    6/Type,12/0,06/20,H12/Terminal Identification
L+2     ZERO   Status Word Pointer,Courtesy Call
L+3     Next Instruction
```

Operation code 20 octal specifies a request for current line status.

Terminal identification is a two-character code that identifies the terminal whose line status is requested.

In response to the MME, the communications subsystem sets the user program status word to nonzero by setting the appropriate status bit. It also inserts the terminal type in bits 0-5 of the VFD instruction. (Terminal type codes are defined under Program Requests Terminal Type.) A bit in the user program status word is set to 1 as follows:

```
Bits 0-28 = 0
       29 = 1  Line connected to another slave program
       30 = 1  Line waiting to connect to slave program
       31 = 1  Line idle
       32 = 1  Line busy
       34 = 1  Line disconnected
       35 = 1  Break received; Break is reset
```

Courtesy Call is not applicable.

Direct Access Output - PPT Input

A user program can send a message to a teletypewriter terminal and in return receive Punched Paper Tape (PPT) input from the terminal. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/Record Pointer,06/21,H12 Terminal Identification
L+2    ZERO    Status Word Pointer,Courtesy Call
L+3    Next Instruction
```

Record pointer is a pointer to the output data block. (This is the same as the record pointer described earlier in this section under Direct Access Output, Then Input.)

Operation code 21 octal indicates output and initiation of paper tape mode.

Terminal identification is a two-character code that identifies the teletypewriter terminal.

The status word pointer is the same as for Direct Access Output, Then Input. The format of the input block is the same as that described in the following section for Accept DAC Paper Tape Input.

In response to this MME, the communications subsystem transmits the user program data to the DATANET processor and informs it that the terminal will now send paper tape input. Prior to the execution of this MME, the user program must have words 0 and 1 of the output data in the proper format in the data buffer.

Word 0 of the data buffer must contain the number of words to be transmitted (including the character count word) and the address of the input buffer which is to receive the first block of PPT input. Word 1 must contain the number of nine-bit characters to be transmitted (a maximum of 312 characters is allowed).

The status word is set to zero when the MME data has been picked up from the slave program and is set to nonzero when the MME request has been completed.

Courtesy Call is used.

Accept DAC Paper Tape Input

After having initiated the paper tape mode by issuing a MME GEROUT with an operation code of 21 (octal), a user program can request that another block of paper tape input be inserted into its data area. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/Record Pointer,06/22,H12/Terminal Identification
L+2    ZERO    Status Word Pointer,Courtesy Call
L+3    Next Instruction
```

Record pointer points to an input buffer which received the block of paper tape input. An input buffer of 31 words is required to accommodate paper tape input from a teletypewriter terminal.

Operation code 22 octal indicates a request for a block of PPT input.

Terminal identification is a two-character code that identifies the terminal.

Status word pointer is the same as for Direct Access Output.

Time Sharing To Direct Program Access Switching

Executable only by TSS, this MME GEROUT function switches a line from the time sharing mode to the Direct Program Access (DAC) mode. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/0,06/25,12/0
L+2    BCI     1,SNUMB
L+3    Normal Return
```

The line connected to the slave program identified by the SNUMB in L+2 is switched from the time sharing mode of operation to the DAC mode. The SNUMB in this calling sequence must be the same as the SNUMB used by the slave program in its inquiry.

Direct Program Access To Time Sharing Switching

This MME GEROUT function switches a line from the Direct Program Access (DAC) mode of operation to the time sharing mode of operation. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/0,06/26,12/0
L+2    BCI     1,SNUMB Number
L+3    Normal Return
```

This calling sequence switches the mode of the line connected to the slave program identified in L+2 from DAC mode to time sharing mode. If the line connected to the slave program was originally a time sharing line, the switch is made and the station identification of that line is put in bits 24-35 of L+1 of the calling sequence. The Time Sharing System is the only program which can issue this MME. The SNUMB in this calling sequence must be the same as the SNUMB used by the slave program in its inquiry.

Generalized Direct Program Access Line Switching

This MME GEROUT is used to switch control of a specified direct access line to another program. The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/Status,06/27,H12/Identification
L+2    BCI     1,Name
L+3    Return
```

The Status entry in this calling sequence means a status word pointer. A nonzero entry in Status points to one word of line status, as follows:

Bit 35 = 1	Break received, line switch request ignored. Break status is reset.
34 = 1	Line no longer connected to system.
32 = 1	Line busy, cannot complete switch.
30 = 1	Line logically disconnected, waiting to connect to slave.
29 = 1	Line logically disconnected, connected to another slave.

Station identification is a two-character code identifying the line to be switched.

The name field in L+2 is the remote inquiry name of the program to which control of the line is switched.

Stop Paper Tape Input

To stop paper tape input prior to sending an error message to the terminal, a DRL STOPPT in the user program causes the Time Sharing System executive to issue this MME GEROUT. The request format is:

```
L       MME     GEROUT
L+1     VFD     18/0,06/30,H12/Terminal Identification
L+2     ZERO    Status Word Pointer, Courtesy Call
L+3     Next Instruction
```

Operation code 30 octal is a request to stop paper tape input.

The terminal identification is a two-character code identifying the teletypewriter terminal.

The status return word is always normal.

Courtesy call is delayed until the communications subsystem determines that paper tape input has stopped.

Extended Input Line Length For DAC Programs

This MME GEROUT function permits a program executing in the Direct Program Access (DAC) mode to use an input line length greater than the standard 80 character line (teleprinter) or 320 words for VIPs and remote computers.

The calling sequence for this routine is as follows:

```
L      MME     GEROUT
L+1    VFD     18/BUF,06/44,H12/ID
L+2    ZERO    Status Word Pointer,Courtesy Call Pointer
L+3    Normal  Return
BUF    ZERO    Number of words to be output, 0
       ZERO    Number of characters to be output, 0
       VFD     9/1,27/Number of characters in the new line length
                Minimum - 4 characters
                Maximum - 408 characters for teletypewriter
                        - 4096 characters for others
```

Operation code 44 octal specifies a request for extended line length.

One MME GEROUT from the .MROUT module provides a maximum of 4096 characters for extended line length for other than teletypewriter terminals. However, practical use is dependent upon possible hardware and software limitations.

For a DATANET 355/6600 processor, this MME GEROUT causes an operation code 36 to be sent to the DATANET 355/6600 processor. The response may be any of the following operation codes (in octal):

125 - Terminal characteristics change acknowledgement.

Bits 0-17 of mailbox word 2 will contain one of the following status codes:

0 - Request Accepted
1 - Illegal Characteristics Type
2 - Illegal Parameter

113 - Line Break - Request unsuccessful

101 - Line Disconnect

This MME GEROUT may cause one of the following aborts:

INVALID MME PARAMETER - The contents of BUF+2 are invalid; i.e., bits 0-8 do not equal 1 and/or the number of characters specified in bits 9-35 is greater than 104 for teletypewriters or is greater than 4096 for VIP's and remote computers.

K1 - INVALID I/O ON DEVICE - The number of words times four is less than the number of characters.

K2 - BAD STATUS RET. PTR.

K4 - INVALID DCW POINTER

K5 - INVALID COURTESY CALL POINTER

RMT TERMINAL RECORD SIZE - Invalid record size

If the terminal is configured on a DATANET 355/6600 processor, the DAC program may receive one of the following status returns as a result of the status information associated with operation code 125.

004 - Request Successful

200 - Illegal Characteristic Type

400 - Illegal Parameter

Remote Status Inquiry For Break/Disconnect Report

This MME GEROUT function, operation code 45, allows a slave program operating in Direct Program Access (DAC) mode to request the break/disconnect status of a line connected to the program and/or to acknowledge previously returned status.

The calling sequence is as follows:

```
L      MME     GEROUT
L+1    VFD     18/ID,06/45,12/0
L+2    ZERO    Status Return Pointer, Courtesy Call Address
L+3    Next Instruction
```

Operation code 45 is a request for break/disconnect status of a terminal and/or to acknowledge previous break/disconnect status for a specified terminal. If the DAC program is acknowledging a previously reported status, the identification field identifies the terminal. If no acknowledgement is being made, this field contains zeros.

This MME GEROUT gives a DAC slave program the ability to request the return of break or disconnect status received for any line connected to the program. If there are no unreported break or disconnect statuses to be returned when .MROUT receives the request, the condition is retained in a manner similar to Remote Inquiry processing.

Return of break acknowledgement or disconnect accepted operation codes to the Front-End Network Processor are delayed until the DAC program connected to the line has acknowledged receipt of the status. Disconnects are acknowledged immediately if the DAC program is aborting.

When .MROUT receives this MME GEROUT, all subsequent status returns are handled as defined for this MME. Therefore, a DAC slave program intending to use this capability should issue the generalized remote status inquiry immediately prior to its first Remote Inquiry MME GEROUT (type 5) upon entering execution.

When operating in main level, a DAC program, using courtesy call to satisfy a remote status inquiry, must not issue a break/disconnect acknowledgement at courtesy call level and then return to main level and issue I/O to that terminal.

The status return pointer specifies the location of a word containing status formatted as follows:

Bits	0-17	- Identification of terminal to which status applies
	18-35	- Status:

1 = Break
2 = Disconnect

When operating in the generalized remote status inquiry mode, these statuses will only be reported via a new or outstanding MME GEROUT, operation code 45. Status returned on I/O that has been terminated because of a break or disconnect is defined as

400 (octal) - I/O aborted

The .MROUT module maintains a table that is used to record outstanding status inquiries. Each two-word entry in the table is formatted as follows:

Word 0	Bits	0-17	Program number
		18-35	Relative location of MME+1
Word 1	Bits	0-17	Status return pointer
		18-35	Courtesy call queue offset

Word 1 contains an offset to the reserved I/O queue (zero if courtesy call is not requested).

When an entry is made to this table, .MROUT obtains an I/O queue if courtesy call is requested in the MME GEROUT call. If courtesy call is requested, any break/disconnect status is reported to the DAC program immediately.

The .MDNET module also maintains the two-word table set up by .MROUT that is used to record the status reporting mode selected by a DAC slave program. Each bit position in this table corresponds to a program number. A bit = 1 indicates that the corresponding program is in the generalized remote status inquiry role. Also, a word in each entry in the terminal line table contains status bits as follows:

```
Bit 0- =1, Break received, slave program not notified
    1- =1, Break returned to DAC program, not acknowledged
    2- =1, Disconnect received, not reported to DAC program
    3- =1, Disconnect returned to DAC program, not acknowledged
    4- =1, Last line for which status was returned
```

"Quick Status" GEROUT

```
L      MME   GEROUT
L+1    VFD   18/0,06/31,H12/TERMINAL ID
L+2    Next Instruction
```

The "Quick Status" GEROUT causes a flag bit to be set in the line table entry for the specified terminal ID. When a future 03 or 04 type GEROUT is performed on the terminal ID, a status bit (Bit 29) in the status return word for the GEROUT is set as soon as the data transfer from the host to the FNP is completed. This "Quick Status" bit is transient in that it is reset when the I/O operation is completed and DNET returns final status.

Switch A Line To Another TSS Copy

```
L      MME   GEROUT
L+1    VFD   18/0,06/33,H12/TERMINAL ID
L+2    ZERO  STATUS WORD POINTER,COURTESY CALL
L+3    Next Instruction
```

Time Sharing is the only program which can issue this MME. This GEROUT changes the program number in the program number field of a line table entry to the program number of the program executing the GEROUT.

GERSTR -- READ FILE IN SYSTEM FORMAT

MME GERSTR makes an entry in the input list of the appropriate service program, enables the execution of that program, and prevents return to the requesting program until the user request has been successfully carried out. Entry to this routine is made from the fault vector as the result of a MME GERSTR.

The calling sequence for this routine is as follows:

```
L      MME     GERSTR
L+1    BCI     1,XXXXXX
L+2    ZERO    Location of first word, Error return address
L+3    ZERO    Transfer address, 0
```

Initially bits 0-23 of the Q-register are unused and bits 24-35 contain the file code of the user's SAVE file.

MME GERSTR restores data saved by MME GESAVE. The data is identified by the file code in bits 24-35 of the Q-register and identified in XXXXXX of L+1. It is read into the location specified in bits 0-17 of L+2. If L+2 contains zeros, the data is stored according to the data supplied in the save operation. If a fatal error occurs on the restore, return is to the address specified in bits 18-35 of L+2. If this field contains zeros, the Restore module aborts the slave program. If the restore is error free, transfer is to the location specified in L+3. If L+3 contains zeros, transfer is to the location specified in the MME GESAVE.

On normal return, the A- and Q-registers contain the following data:

A-register:	Bits	0-17	Loading origin
		18-35	Number of words
Q-register:	Bits	0-17	Transfer address
		18-35	Not used

If an error occurred and the error return address is specified in L+2, the transfer is to that address and bits 18-35 of the Q-register contains one of the following error codes:

Code	Definition
52	I/O limits error
53	I/O error
54	No PAT for CALL/SAVE
55	Bad device for CALL/SAVE
56	Non-random file
60	Checksum error
61	Lower GERSTR origin used
65	Improper call

If the error return address (bits 18+35 of L+2) contains zeros, the program is aborted.

MME GERSTR should not be used during a courtesy call.

GESAVE -- WRITE FILE IN SYSTEM FORMAT

MME GESAVE makes an entry in the input list of the appropriate service program, enables execution of that program, and prevents return to the requesting program until the save request has been successfully carried out.

Entry to this routine is made from the fault vector as the result of a MME GESAVE in a user program.

The calling sequence for this MME is as follows:

```
L       MME     GESAVE
L+1     BCI     1,XXXXXX
L+2     ZERO    Location of first word, Number of words
L+3     ZERO    Transfer address, Loading increment
L+4     Return
```

If an error return is desired, the error return address must be specified in bits 0-17 of the Q-register. If an error occurs during the save, control is returned at this address. The file code of the save file is defined in bits 24-35 of the Q-register.

L+2 (bits 0-17) specifies the location of the first word to be saved and the number of words to be saved (bits 18-35) on the file specified in bits 24-35 of the Q-register. This file can be tape or random disk. The XXXXXX field of L+1 uniquely identifies this data. Bits 0-17 of L+3 contains an address to which control is transferred when the saved file is restored. This address is used only if the MME GERSTR calling sequence contains zeros in the corresponding transfer address word. The loading increment (bits 18-35 of L+3) is a value to be subtracted from the location of the first word and the transfer address. It is used in reloading the data via a MME GERSTR. Bit 15 of the Program Switch Word must be set (=1) prior to the first MME GESAVE.

Do not copy a save file to a new file of a smaller size and then try to save additional data onto the new smaller file (a TSS copy will always make the new file smaller if not all of the original file is used).

If an error occurred and an error return address is specified, return is to that address, and bits 18-35 of the Q-register contain one of the following error codes:

Code	Definition
52	I/O limits error on save
53	I/O error save
54	No PAT for CALL/SAVE
55	Bad device save
56	Non-random file
57	Save file is full
60	Checksum error
62	Zero word count
115	Impermissible permanent file write

If the error return address (bits 0-17 of the Q-register) contains zeros, the program is aborted.

Master mode programs (e.g., ALOC, NPS, GRTS, GESNAP) must not use the MME GESAVE instruction. MME GESAVE should not be used during courtesy call.

GESECR -- PASSWORD ENCRYPTION

MME GESECR is used by PRIVITY programs to encrypt a password that has been loaded into the AQ Register. The encrypted password is returned in the AQ Register.

The calling sequence for the MME GESECR is as follows:

```
L-1  LDAQ     PASWD    Password to be encrypted in AQ
L    MME      GESECR
L+1  Return            Encrypted password in AQ
```

All programs without PRIVITY that attempt to execute this MME will be aborted with error code of 121, INVALID MME.

GESETS -- SET SWITCH REQUEST

MME GESETS is used to set bits in the program switch word according to corresponding bits in the Q-register. For each bit position of the Q-register that contains a 1, the corresponding bit of the switch word is set to 1. For each bit position of the Q-register that contains a 0, the corresponding bit of the switch word is not changed. The resultant setting of the switch word is returned in the Q-register (see Section II).

Entry to this routine is made from the fault vector as the result of a MME GESETS instruction. The calling sequence for MME GESETS is as follows:

```
L      MME      GESETS
L+1    Return

Initially:      C(Q) = 36-bit mask

Results:  C(Q) = Contents of switch word
```

GESNAP -- SNAPSHOT DUMPS

MME GESNAP is used to obtain printouts of selected parts ("snapshots") of main memory during the debugging of a program. The snapshot dump produced is written on the P* file, if this file is assigned to SYSOUT or to a printer. Any other assignment of P* results in the snapshot dump being written on the execution report.

Entry to this routine is made from the fault vector as the result of the MME GESNAP. The calling sequence is as follows:

```
L      MME      GESNAP
L+1    VFD      18/A,2/P,1/S,15/N
L+2    Return
```

```
Where:  A - Starting location of the snapshot dump.  A may be
            symbolic.  If A = modulo 8, N+7 is truncated to modulo 8.
            If A is not modulo 8, N+14 and A are truncated to modulo
            8.
        P - Panel indicator specifying portions of panel to be dumped:
            00 = Panel, EIS registers (EIS processor), and memory
            01 = No registers, memory only
            10 = Panel and EIS registers only
        S - Slew indicator
            0  = Slew one line before snapshot dump
            1  = Slew to top of page before snapshot dump
        N - Number of words to be snapped
        N - =0, Memory beginning at A will be dumped.
```

MME GESNAP may not be used during courtesy call.

GESNUM -- SUPPLY SEQUENCE NUMBER

MME GESNUM is used to assign system-generated SNUMB's to slave programs. Each system program which generates SNUMB's has a unique identifier, as follows:

T = Time Sharing
Z = Spawn

System date and time and a random incrementing factor are used to generate the SNUMB's. The .CRSEQ word and the SNUMB table prevent generation of duplicate SNUMB's. At midnight, the GESNUM routine is reset via a store zeros instruction in .CRSEQ.

Entry to this routine is made from the fault vector as a result of the MME GESNUM.

The calling sequence for this routine is as follows:

```
L      MME     GESNUM
L+1    Return

Initially:  C(Q) required

                 Bits  0-29   Must be zero
                      30-35   Identifier
```

On return, the Q-register contains:

```
Bits  0-23   Generated SNUMB
     24-29   Identifier
     30-35   17 (octal)
```

The only identifiers currently assigned are T and Z.

GESPEC -- SPECIAL INTERRUPT REQUEST

MME GESPEC presets conditions so that an I/O request will be accepted, but the I/O will not be started until a special interrupt occurs on the device. Alternatively, this routine can nullify the effects of a previous MME GESPEC request and delete any unstarted I/O request from the queue.

Entry to this routine is made from the fault vector as the result of a MME GESPEC. The standard MME call input parameters are used. When the MME GESPEC is executed, the Q-register must have the pertinent file code in bits 24-35. In addition, if the effects of a previous MME GESPEC are to be nullified, bit 0 of the Q-register must be 1.

The calling sequence for this routine is as follows:

```
L      MME     GESPEC
L+1    Return

Initially:       C(Q) required

            Bits  0   = 0 MME GESPEC
                      = 1 Nullify affect of previous MME GESPEC
               1-23   Must be zero
              24-35   File code
```

As a result of the MME GESPEC the A-register will contain an indication of the response:

0 - Requested action taken

1 - No action taken; file code not in PAT

2 - No action taken; GESPEC illegal for specified device

MME GESPEC is allowed for the card reader, card punch, and printer.

GESYOT -- WRITE ON SYSOUT

MME GESYOT is used to transmit output records to the SYSOUT collector media for subsequent printing or punching. SYSOUT collects these records, which may constitute several different reports for each concurrent activity.

The output files P* (GMAP listing), C* (binary cards from GMAP), and K* (COMDK cards) are assigned to SYSOUT, unless otherwise assigned by control cards. Additional files may be assigned to SYSOUT by the $ SYSOUT control card. Files assigned to SYSOUT may not be processed with MME GEINOS. The MME GESYOT provides the control necessary for such a shared device.

Entry to this routine is made via the fault vector as a result of the MME GESYOT instruction.

Normally, SYSOUT is referenced only by the File and Record Control function, where the user references files via a file control block rather than as a specific device. The calling sequence is as follows:

```
L       MME      GESYOT
L+1     ZERO     FCB,0
L+2     Return
```

FCB is the location (word 0) of the file control block for the file containing the records to be transmitted. SYSOUT uses various fields of the file control block to determine the location and quantity of the data to be written. Bit 18 of each logical record control word set indicates to the MME GESYOT processor that the record has been processed. This bit must be reset before attempting another MME GESYOT.

NOTE: The data buffer used by MME GESYOT is in standard system format; however words 2 and 3 are not used and are skipped by the MME processor.

In using MME GESYOT, the status must be monitored to ensure that the I/O is completed before attempting to re-use the buffer.

Only three remote terminals per job may receive output via MME GESYOT. Only nine report codes per activity are allowed. The total record count for all reports of an activity cannot exceed the user defined or default allocation limit.

MME GESYOT should not be used in a courtesy call.

Printing Files Via SYSOUT

A user program may request that a mass storage file of print or punch records be passed to SYSOUT for processing as soon as a device is available. These records must be written in standard system format via File and Record Control or MME GEINOS. The request is made by the following MME GESYOT calling sequence:

```
L       MME      GESYOT
L+1     BCI      1,0000FC
L+2     BCI      1,AAAAAB
L+3     Denial return
L+4     Normal return
```

Where: FC - File code of the user's temporary mass storage file

AAAAA - Five-character field which may be used to provide a unique output banner. If no special banner is required, this should be zeros. User's SNUMB will be used for a banner.

B - Type of output:

```
B = 1   Punched output        (media codes 1 or 2)
B = 2   BCD printed output
B = 3   ASCII printed output
```

The following conditions result in an abort of the user's program. The abort code appears on the output.

- o User's file is a system file, secondary file, or a released file. Abort code is 143 (octal).
- o User's file is not a mass storage file. Abort code is 143 (octal).
- o User's file and SYSOUT space are on devices having different sector sizes. Abort code is 143 (octal).
- o User's file code could not be found in the program's Slave Service Area (SSA).

The following conditions cause a denial return. A return code is given in the lower half of the A-register.

- o The SYSOUT backdoor file is not configured. The denial code is 005.
- o The backdoor file is full; i.e., file space is exhausted. The denial code is 001.
- o A denial return is made for ASCII files when there is no ASCII printer configured. The denial code is 005.

Control is returned to the user program on a denial return. A denial return code of 001 indicates that a subsequent request may be processed successfully, and the user may want to generate a loop on this status return to attempt the MME GESYOT again. A denial return code of 005 indicates that a subsequent request will not be processed successfully, and the user may wish to relinquish control.

The file should be written in standard system format, but block serial numbers are not necessary. The file should contain an End-of-File mark at the end of the data, or the data should fill the file. All records must contain media codes, and print records should contain printer slew codes. Punched output can only be media codes 1 or 2; any other media codes will be ignored.

The file is processed in sequence without regard for the report codes in the record control words and is released after processing. Thus the file cannot be reprocessed, even if reprocessing is requested.

The backdoor file is a short system file in which the PAT of the user file is recorded for subsequent access by SYSOUT. When the MME GESYOT for printing files is received, SYSOUT writes the PAT and banner information to the backdoor file and indicates to the SYSOUT Report Writer that a backdoor job is ready for processing.

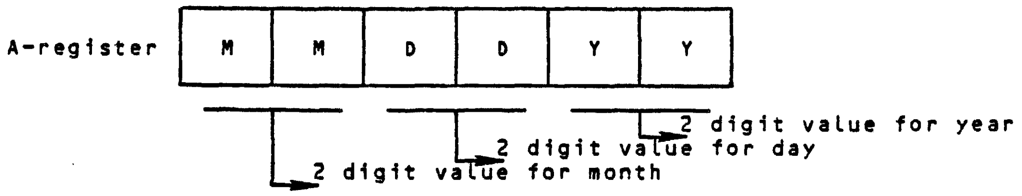
The SYSOUT report writer normally produces the identification banner upon encountering the $ IDENT card on J* with record control word of the form 18/length, 18/050374 (18/050774 for ASCII). If such a record is recorded as the first logical record of the user's file, the report writer accepts it and produces a banner for the backdoor job.

GETIME -- DATE AND TIME OF DAY REQUEST

MME GETIME provides the requesting program with the date and the time of day.

Entry to this routine is made from the fault vector as the result of a MME GETIME instruction. Return is made to the requesting program with the date (in the form of MMDDYY in binary-coded decimal) in the A-register and the time of day right-justified in the Q-register. Time of day is expressed in the number of 1/64 millisecond pulses past the previous midnight.

Format of the A-register is as follows:



The calling sequence is as follows:

```
L      MME     GETIME
L+1    Return
```

The time of day returned to the program is that indicated by the simulated clock for the processor on which the user is currently executing. For a user program in a uniprocessor environment, the value is correct to within 20 milliseconds. In a multiprocessor environment, the value may be off by as much as several seconds. Consequently, the use of MME GETIME is not recommended for timing studies.

GEUSER -- USER-SUPPLIED MME

MME GEUSER gives the user installation the ability to add its own MME for a specified use. Through this entry, an installation may write its own module (.MUMME) tailored to its own needs.

Entry to the .MUMME routine is made through the fault vector as a result of a MME GEUSER instruction in a user program.

The calling sequence for this routine is specified by the user installation.

The user installation is responsible for maintaining this module. The use of MME GEUSER will cause a slave program to be aborted by GCOS (I1 abort - improper MME address) if the user module is not supplied.

The MME GEUSER may not be used during courtesy call.

GEWAKE -- CALL-ME-LATER

MME GEWAKE is used in a program to suspend processing for a specified time and then become a candidate again for processor time. Candidacy occurs when the time interval, in milliseconds, has elapsed, when any outstanding I/O for the program has terminated, or when the program has been requested by some other program.

Entry to this routine is made from the fault interrupt vector as the result of the MME GEWAKE instruction.

The calling sequence for this routine is as follows:

```
L      MME     GEWAKE
L+1    Return
```

Initially, the Q-register contains the time interval in 1/64 millisecond increments, after which this program is enabled. The value is right justified in the register.

The MME GEWAKE should not be used during a courtesy call. There is no minimum or maximum time interval. However, if the time interval is greater than three seconds, the program may be swapped out of memory.

GEXLIT -- TRANSLITERATION TABLE ACQUISITION

MME GEXLIT is used to bring into the user's memory space the transliteration tables required for output transliteration processing. The calling sequence for this instruction is as follows:

```
L       MME     GEXLIT
L+1     VFD     18/list pointer,9/0,9/number of entries in list
L+2     ZERO    0,location
```

Where: list pointer - Points to list of table names (BCD) that the user wants loaded.

number of entries in list - Number (decimal) of entries in the list of table names.

location - Relative location at which the first table is to be loaded.

The tables are loaded into the user's memory space in the order that they appear in the table name list.

The tables returned by MME GEXLIT are set up to be used with EIS instructions. The table names are defined in the following table:

Transliteration From - To	Legal Table Names	Size in Words
GBCD, ASCII	U.GTOA	16
GBCD, EBCDIC	U.GTOE	16
GBCD, HBCD	U.GTOH	16
ASCII, GBCD	U.ATOG	128
ASCII, HBCD	U.ATOH	128
ASCII, EBCDIC	U.ATOE	128
EBCDIC, GBCD	U.ETOG	128
EBCDIC, ASCII	U.ETOA	128
EBCDIC, HBCD	U.ETOH	128
HBCD, GBCD	U.HTOG	16
HBCD, EBCDIC	U.HTOE	16
HBCD, ASCII	U.HTOA	16
GBCD, IBMCC	U.XLTA	16
GBCD, IBMCT	U.XLTB	16
GBCD, IBMFC	U.XLTC	16
GBCD, IBMFT	U.XLTD	16
GBCD, GE225C	U.XLTE	16
GBCD, GE225T	U.XLTF	16
GBCD, IBMEL	U.XLTG	16
IBMCC, GBCD	U.XLTH	16
IBMCT, GBCD	U.XLTI	16
IBMFC, GBCD	U.XLTJ	16
IBMFT, GBCD	U.XLTK	16
GE225C, GBCD	U.XLTL	16
GE225T, GBCD	U.XLTM	16
IBMEL, GBCD	U.XLTN	16

SECTION V

SLAVE PROGRAM - GCOS COMMUNICATIONS

SLAVE PROGRAM PREFIX

The slave program prefix resides in the first 64 words of slave memory and is shown in Figure 5-1. The slave program prefix contains the following areas:

- o Fault Vector
- o GCOS Reserved Areas
- o General Loader Communication Region
- o GCOS Initiation and Termination Sequence
- o Register Safe-Store Areas

Prior to loading any activity, the entire slave program prefix, with the exception of words 26 through 29, is set to zero.

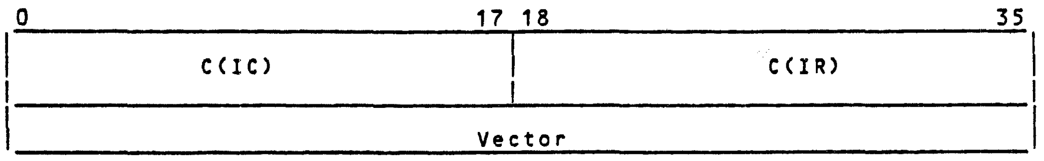
NOTE: Figure 5-1 describes locations by both octal and decimal numbers, however locations are referenced by decimal numbers in the text.

FAULT VECTOR

The fault vector comprises the first 14 words (Words 0-13) of the slave program prefix. These words are used in pairs, one for each fault trap (with the exception of the Lockup fault).

Word 0 and 1	-	Illegal Procedure fault Command fault Parity (Time Sharing System only)
2 and 3	-	Memory fault Lockup (Time Sharing System only)
4 and 5	-	Fault tag modifier Operation not complete (Time Sharing System only)
6 and 7	-	Divide check fault
8 and 9	-	Overflow fault (including one caused by truncation condition)
10	-	Lockup fault, Parity fault, Operation not complete fault
11	-	Location of abort and the reason code of last abort (see Appendix A)
12 and 13	-	Derail instruction

Each word-pair has the following format:



In the first word, GCOS software stores the contents of the Instruction Counter (IC) and Indicator Register (IR) as they were when the fault occurred.

GCOS software performs its fault handling in the following manner:

1. For Lockup, Parity and Operation Not Complete (ONC) faults, the IC and I are stored in location 10 and the program is aborted.

2. For Divide Check, Derail, and Overflow faults, processing is as follows:

   a. The IC and I are stored in the first word of the fault vector.

   b. If the second word of the fault vector contains zero, the program is aborted.

   c. If the second word is nonzero, GCOS software transfers control to that word after restoring all registers to their status at the time of the fault.

   d. At program load time the fault vectors are left zero if the $ OPTION control statement says NOSETU, or if a language setup routine is not defined. In the event of a fault and this $ OPTION condition, the second vector word, containing zeros, would result in a program abort.

      If COBOL-68 is specified in the control statement, the second word of the fault vectors for divide check and overflow are set to subroutines that return immediately to the user's program. Under this condition, the GCOS operating system does not process the fault.

      If COBOL-74 or UFAS is specified in the control statement, the IPR fault vector is used to determine whether the application is running in Time Sharing.

      If FORTRAN or JOVIAL is specified in the control statement, .FSTU for batch or .FTSU for Time Sharing:

      1) Places the address of the Logical File - File Control Block Table in word 21 in the Slave Program Prefix.

      2) For a link job with DEBUG requested in link 0, places the DEBUG Table address in word 13 in the Slave Program Prefix.

      3) For a low-load job, places the entry point and an indicator bit in word 20 of the Slave Program Prefix.

      4) Calls .FLTPR to initialize for fault processing.

A separate FORTRAN subroutine also exists which places zeros in words 7 and 9 in the Slave Program Prefix for divide check and overflow faults. If no FORTRAN library subroutines are used, this routine is loaded which causes divide check and overflow faults to abort.

The .FLTPR routines return to .FSTU. .FSTU zeros all index registers and transfers to the user's program.

GCOS software initially clears the first and second word to zero. The following two examples illustrate how a slave program might use word 2.

1. To ignore Divide Check, Derail and Overflow faults, the second word (n+1) is set to:

```
RET     n
```

2. To process a fault with a routine at location FLT, the second word is set to:

```
TRA     FLT
```

3. For Illegal Procedure (IPR), Store (Memory), or Fault Tag Modifier fault, GCOS software processes the fault as follows:

   a. GCOS software stores the contents of the IC and I as they were at the time the fault occurred.

   b. If the second word contains zero, the program is aborted.

   c. If the second word is nonzero, GCOS software restores all registers (except the timer) to their status at the time of the fault and transfers control to that word. GCOS software also stores the fact that the fault occurred.

   d. On subsequent faults of these types, GCOS software will continue to process the fault as previously described, if the first (even) word has been zeroed. If the first word was not zeroed, GCOS software aborts the program.

C (BAR) + Word:

Decimal	Octal		
0	0	Illegal Procedure and Command	
1	1		
2	2	Memory	
3	3		
4	4	Fault Tag	
5	5		
6	6	Divide Check	Fault
7	7		Vector
8	10		
9	11	Overflow	
10	12	Op Not Complete, Lockup, Parity	
11	13	Loc. of Abort, Reason Code of Last Abort	
12	14		
13	15	Derail	
14	16	Control Address After Check Point Recov.	
15	17	Address No. of FCB in Activity	
16	20	File and Record Control Switch Word	Reserved for
17	21	GELBAR Timer Setting	GCOS
18	22	IC and I for GELBAR	
19	23	Fault Vector for GELBAR	
20	24	Address of Entry Lowload Flag	
21	25	LGU Table Destination Code Table	General
22	26	History Reg. Buf. R (Retry Bit)	Loader
23	27	Wrapup Address GELOOP Address	Communication
24	30		
25	31	Accumulated Fault Status	
26	32	MME GECALL	
27	33	Name	GCOS initiation
28	34	0	and Termination
29	35	0	Sequence
30	36	Activity No., Job Sequence No.	
31	37	Loader Relative Load Limits	
32	40	X0 X1	
33	41	X2 X3	
34	42	X4 X5	
35	43	X6 X7	Register
36	44	A-Register	Safe-Store
37	45	Q-Register	Area 1
38	46	E Reg.	
39	47	Timer Register	
40	50	Main level registers saved here when	Register
.	.	activity is in courtesy call. See words	Safe-Store
47	57	40-47 (octal) for format.	Area 2
48	60	EOF Reissue Buffer	
49	61		
.	.		
.	.		
53	65		Reserved
54	66	Image of Cols. 13-72 of	for
.	.	$ IDENT Card	GCOS
.	.		
63	77		

Figure 5-1. Slave Program Prefix Layout

GCOS RESERVED AREA

Words 14 through 19 and word 22 are reserved for GCOS software. Word 14 contains the address to which control is returned after a checkpoint recovery.

Word 15 contains information about the File Control Blocks (FCB) in the activity. Bits 0 through 17 contain the address of the first FCB's in the activity. This address is used by a MME GEROLL instruction to find files to be repositioned. Bits 18 through 35 show the number of FCB's in the activity and is used by File and Record Control to update the linkage of the FCB's on any Open or Close calls.

Word 16 is reserved for use as a File and Record Control switch word. Bit 18 is used as an interface between .GCLOS and .GR200 regarding output to SYSOUT files.

Words 17 through 19 are used by the MME GELBAR processor to contain the GELBAR processor time (word 17) until the timer runs out; the instruction counter relative to the new base where execution is to be continued and the desired indicator register setting (word 18); and the MME GELBAR fault vector (word 19).

Bits 0-16 of word 22 contain a pointer to the processor History Registers buffer. Bit 17 is a flag bit for the RETRY function. Bits 18-35 are reserved for GCOS software usage.

GENERAL LOADER COMMUNICATIONS

Words 20, 21, 23, and 25 are used for General Loader communication.

Word 20 has the following format:

Bits 0-17	Address of entry to main program
18	= 0 Normal (Highload)
	= 1 Lowload
19-35	Not used

Word 21 contains the following:

Bits 0-17	Address of logical unit table
18-35	Address of destination code table

Word 21 is also used by the file editor.

Word 23 is used as an address of the wrapup routine (bits 0-17) and the return address after a MME GELOOP timeout (bits 18-35). This word has the following format:

Bits 0-17	Address of Wrapup routine
18-35	GELOOP transfer address

Word 25 has the following format:

Bits 0-17	BAR from last MME GELBAR
18	Not used
19	Fault bit
20	Reserved for GCOS software
21	Overflow bit
22	Exponent overflow bit
23	Exponent underflow bit
24	IPR fault bit
25-28	Must be zero
29	Divide check fault bit
30-35	Fault type

Each Overflow, Exponent Overflow, Exponent Underflow, and Divide Check fault handled by GCOS software causes the fault type to be stored in bits 30-35 and appropriate bits to be set in bits 19-29 of word 25 as indicated below:

Bit Position		Fault
19	= 0	I/O interrupt occurred when MME GELBAR was in effect
	= 1	FAULT occurred within a MME GELBAR
20		Reserved for GCOS software
21		Overflow
22		Exponent overflow
23		Exponent underflow
29		Divide check
30-35		Fault type inside MME GELBAR:

0	MME
1	Memory
2	Fault tag
3	Command
4	Derail
5	Lockup
6	Illegal procedure
7	Operation not complete
8	Overflow
9	Divide check
10	Timer runout
11	Parity

If the user supplies his own fault handling routines, bits 21, 22, 23, and 29 of word 25 are not altered by GCOS software when any fault under MME GELBAR occurs. MME GELBAR allows fault processing responsibility to be placed on the user. When a fault occurs under MME GELBAR, the FALT module transfers control to the user's GELBAR processing routine. Before transferring to the user's fault processing routine, bit 19 and the fault type are "ored" into word 25.

When an Illegal Procedure (IPR) fault occurs, bit 6 of the fault register is placed in bit 24 of word 25 of the slave program prefix. This allows the programmer to determine if the IPR fault was caused by the execution of an EIS instruction with bad data. If a program aborts with an IPR fault, the fault code is changed from 31 (octal) to 163 (octal) if bit 6 of the fault register is a 1. The abort message for code 163 (octal) is ILLEGAL EIS DATA.

A slave program that handles its own IPR faults can also test slave prefix word 25, bit 24 for the Illegal EIS Data condition.

GCOS Initiation And Termination Sequence

Words 26 through 29 are used for the initiation and termination sequence. To load the General Loader, the Allocator sets these words as follows:

Word	Contents
26	MME GECALL
27	BCI 1,GELOAD
28	ZERO
29	ZERO

Activity Identification

The activity number and job sequence number are contained in Word 30.

Bits	0-5	Activity number (octal)
	6-35	Job sequence number (BCD)

General Loader Relative Load Limits

Word 31 has the following format:

Bits	0-17	Lower limit
	18-35	Upper limit

The lower limit is the lowest allowable address of the activity to be loaded by the General Loader, relative to the origin of the loader.

The upper limit is the highest allowable address of the activity to be loaded, relative to the origin of the loader.

This word is set by GCOS software prior to loading the General Loader.

Upon completion of loading, the General Loader sets Word 31 of the object memory to the next available lower and upper memory addresses.

Register Safe-Store Areas

There are two register safe-store areas. Both are blocks of eight words each and are used by GCOS software for safe-storing the slave program's registers.

Words 32 through 39 are used for safe storing slave program registers as follows:

Word (octal)	Content	
	Bits 0-17	Bits 18-35
40	Index Reg. 0	Index Reg. 1
41	Index Reg. 2	Index Reg. 3
42	Index Reg. 4	Index Reg. 5
43	Index Reg. 6	Index Reg. 7
44	A-register	
45	Q-register	
46	E-register (Bits 0-7)	
47	Timer Register	

Words 40 through 47 are used to save main level registers while the activity is in courtesy call.

GCOS Reserved Area

Words 48-63 are also reserved for GCOS software use. The I/O Supervisor (IOS) uses words 48-53 to store Data Control Words (DCW's) on End-Of-File (EOF) processing. A portion of the $ IDENT card (columns 13-72) is placed into words 54 through 63.

JOBS IN ALLOCATION

Site-written privileged slave programs must scan Peripheral Allocator tables for information about jobs in allocation.

SLAVE PROGRAM PREFIX AT TERMINATION

When an activity aborts, seven words of the slave program prefix can be used to analyze the abort. These words and their application in analyzing various types of aborts are described in the following paragraphs.

First Abort

In the initial abort, words 11, 13 and 18 (octal 13, 15 and 22) take on the following meanings. Words 12, 19, 28 and 29 (octal 14, 23, 34 and 35) are reserved.

```
Word 11 (13), Bits  0-17  Instruction counter (IC) indicating location of
                          first termination
                   18-20  Must be zero
                   21-23  Type of abort:
                             0 - MME GEFINI
                             1 - GCOS/hardware fault
                             2 - MME GEBORT
                             3 - ABORT input from console
                   24-35  Abort code (see Appendix A)

Word 13 (15)   - Contents of .SNTRY word at time of fault

Word 18 (22)   - IC and I of last termination
```

Wrapup Abort

If the abort occurs during wrapup, word 12 (octal 14) contains a copy of word 11 (octal 13) and word 19 (octal 23) contains a copy of word 18 (octal 22). Word 13 (octal 15) is the same as for first abort. Words 28 and 29 (octal 34 and 35) are reserved.

If a slave program aborts in an SSA overlay the following action is taken:

A block of 510 words starting at location .SSA is moved into the slave program starting at location 100 (octal). This address is recorded in bit locations 0-17 of word 30 (octal) of the SPP. Wrapup is not honored. Finally a snap of this block with header "LOW" is provided.

Exception Processing Abort Data

Under the following abort conditions, the Exception Processing program stores information in words 28 and 29 (octal 34 and 35) of the slave program prefix.

1. The operator responded A (abort) to an Exception Processing program error message.

2. The error for which the Exception Processing program is aborting the user program is an IOM error (channel or IOM central).

3. The console operator requested rollback.

The two words of data are:

```
Word 28 (34), Bits  0-17  Status return pointer
                   18-23  =77 (Exception Processing flag)
                   24-35  File code
```

Word 29 (35), Bits	0	=1, Status loaded
	1	=1, Power off
	2-5	Major status
	6-11	Substatus
	12-15	Reserved
	16	=0, Terminate Interrupt =1, Marker Interrupt
	17	=1, Exception Processing program could not recover
	18-20	IOM channel status
	21-23	IOM central status
	24-29	=77, Rollback was requested
	30-35	Device command

GCOS Abort

On a GCOS abort, the existing abort information appears in the slave program prefix and words 28 and 29 (octal 34 and 35) are loaded with the IC and I of the abort and the abort reason code respectively.

PROCESSOR HISTORY REGISTERS

The processor contains 32 history registers (16 used by the control unit and 16 used by the operations unit) which perform a hardware trace function. These registers are for the use of Honeywell Field Engineering representatives in analyzing hardware faults. These registers record the state of the processor hardware for the last 16 steps of its operation. Data is written into a register at the end of each operand or address (for instruction fetch) cycle in the control unit and at the end of the last cycle of the operations unit.

Any of the following faults cause the history registers to be stored in an error record buffer, which contains the history registers and related operating data.

- o Operation Not Complete
- o Lockup
- o Parity
- o Command
- o Illegal Procedure
- o Store

Slave programs may have access to the error record buffer by placing the even address of a buffer in bits 0-17 of word 22 of the slave program prefix. Although the error record buffer size may vary, it will normally be less than 150 words in length, and this should be considered in establishing the buffer size. This slave program buffer will be loaded with the error record data when one of the preceding faults occurs. This may vary as follows:

1. For a program running under MME GELBAR, this is before a return to word 19.

2. For a user processed fault, this is before return to the fault vector in the slave program prefix.

3. For an aborting user program, this is before return to wrapup and before a snapshot dump.

The format of word 22 of the slave program prefix is:

Bits 0-16	Address of buffer in which error record is to be stored (must start even) divided by two. Least significant bit is removed. Address is relative, but if it is illegal it is ignored.
17	= 0, Instruction will be retried. = 1, Instruction will not be retried (unless in master mode).
18-35	Not used.

Programs using trace packages and other functions which generate faults should consider the following:

1. The processing of history registers in the fault processor module can add a lot of unnecessary time to the programs.

2. If faults are expected and the history registers are not wanted, bits 0-17 of word 22 of the slave program prefix should be set to 1. This prevents the addition of extra steps to the fault handling function, and no instruction retry is performed. This has no effect if the fault occurs in master mode.

Figures 5-2 through 5-4 represent the configuration of the history registers for the Series 60 Level 66, 6000, and DPS 8 systems.

OCT	0 56 8 9 11 12 17 18 23 24 27 29 30 35				
00	Size	SR-ID	Reserved		Type (13)
01	LAL or Zeroes	Program #	Retry Ctr.	CPU #	Fault Code
02	Upper Limit Res	BER	BAR		
03	Faulted Instruction				
04	Reason Code	Sequence Number			
05	Reserved	Operand Address			
06 07	Operand Pair in Error				
10	X0	X1			
11	X2	X3			
12	X4	X5			
13	X6	X7			
14	A				
15	Q				
16	Exponent Register				
17	Timer Reg.				
20	Fault Register				
21	Time of Day (.CRDAT+1+.CRTOD)				
22	Mode Register				
23	Mode Register Extension (CACHE)				
24	SNUMB				
25	Configuration Switches (RSW 1)				
26	Configuration Switches (RSW 2)				
27	MBA	MBB			
30	Instruction Counter	Indicators			
31	Date (mmddyy)				
32 71	Control Unit History Registers				
72 131	Operations Unit History Registers				
132 171	Decimal Unit History Registers (Series 6000 EIS and Series 60 Processors Only)				
172 201	Pointers and Length (Series 6000 EIS and Series 60 Processors Only)				
202 211	Address Registers (Series 6000 EIS and Series 60 Processors Only)				

Figure 5-2. History Register Configuration For Series 60 (Level 66)/6000 Processors

HISTORY REGISTER BUFFER

```
OCT  DEC 0     5 6  8 9  11 12     17 18    23 24   27 29 30   35
00    0  |        Size              |    06          |    TYPE      |
         |                  |       |B|B|B|  |T|W|U|RETRY|CPU|              |
01    1  |   LAL or ZERO    | PRG # | | | |  | | | | CTR |  # | Fault Code  |
         |                  |       |L|A|E|  |L|R|R|     |    |             |
02    2  |      Upper Limit          |              BAR                     |
         |                                                                  |
03    3  |               Faulted Instruction                                |
04    4  |    Reason Code                         Sequence Number           |
         |C|S|I|                  |P|L|                                     |
05    5  | | | |      Reserved    | | |          Operand Address            |
         |P|P|P|                  |R|C|                                     |
06    6  |                                                                  |
         |                Operand Pair In Error                             |
07    7  |                                                                  |
10    8  |        X0              |            X1                           |
11    9  |        X2              |            X3                           |
12   10  |        X4              |            X5                           |
13   11  |        X6              |            X7                           |
14   12  |        A                                                         |
15   13  |        Q                                                         |
16   14  |        Exponent Reg                                              |
17   15  |        Timer Reg                                                 |
20   16  |        Fault Register                                            |
21   17  |        Extended Fault Register                                   |
22   18  |        Time of Day (.CRDAT+1)+(.CRTOD)                           |
23   19  |        Reserved                                                  |
24   20  |        Mode Register                                             |
25   21  |        Mode Register                                             |
26   22  |        SNUMB                                                     |
27   23  |        Configuration Switches (RSW1)                             |
30   24  |        Configuration Switches (RSW2)                             |
31   25  |        MBA             |            MBB                          |
32   26  |        IC              |            I                            |
33   27  |        Date (mmddyy)                                             |
         |                                                                  |
34   28  |        Control Unit                                              |
         |        History Register                                          |
         |                                                                  |
73   59  |                                                                  |
74   60  |                                                                  |
         |        Pointer and Length Registers                              |
103  67  |                                                                  |
104  68  |                                                                  |
         |        Address Registers                                         |
113  75  |                                                                  |
```

Figure 5-3. History Register Configuration For DPS 8/20-44 Processor

HISTORY REGISTER BUFFER

OCT	0 – 56 – 8 9 – 11 12 – 17	18 – 23	24 – 27 – 29	30 – 35
00	Size	SR-ID	Reserved	Type (13)
01	LAL or Zeroes | Program #	| | | | |	Retry Ctr. | CPU #	Fault Code
02	Upper Limit | Res | BER	BAR		
03	Faulted Instruction			
04	Reason Code	Sequence Number		
05	| Reserved | | | | | | |	Operand Address		
06–07	Operand Pair in Error			
10	X0	X1		
11	X2	X3		
12	X4	X5		
13	X6	X7		
14	A			
15	Q			
16	Exponent Register			
17	Time Reg.			
20	Fault Register			
21	Time of Day (.CRDAT+1+.CRTOD)			
22	Mode Register			
23	Mode Register Extension (CACHE)			
24	SNUMB			
25	Configuration Switches (RSW 1)			
26	Configuration Switches (RSW 2)			
27	MBA	MBB		
30	Instruction Counter	Indicators		
31	Date (mmddyy)			
32–231	64 CONTROL UNIT REGISTERS			
232–431	64 COMBINED OPERATIONS UNIT/DECIMAL UNIT REGISTERS			
432–441	P/L			
442–451	AR			

Figure 5-4. History Register Configuration For DPS 8/52-70 Processor

SECTION VI

ALLOCATION AND TERMINATION

ALLOCATION

When a new job is received, the job control file, containing all control cards, is scanned to determine whether the job deck was made up correctly and which, if any, magnetic tape reels or printer forms should be mounted. During this scan, the number of executions and number of $ BREAK cards is accumulated for later decision making. The worst case is determined for memory requirements and magnetic tape needs; processor time and output line limits are accumulated. These values are then optionally used to set urgency for the job and to gauge the 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 these is true. If the job is lengthy or requires an unusually large amount of memory, processor time, or output lines and is not an urgent job, it is bypassed and the operator is informed of the bypass.

The first time an activity is considered for allocation, the job control 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. These detail entries 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 accomplished by matching a job's peripheral needs with the available resources described in the System Configuration Tables (SCT), until all needs are satisfied. During this process a Peripheral Assignment Table (PAT) is generated. The PAT allows a user-specified file code to be associated with an entry in the SCT and, at I/O time, with a specific piece of peripheral hardware.

If the job has a .CRNS1 (hardcore table) status of 6, 11, or 12, the Peripheral Allocator is attempting to ascertain core status and one of the following messages is given:

.CRSN1 Status	Message
6	WAITING CORE
11	IN CORE
12	SWAPPED

followed by the message

RE-ENTER STATS FOR MORE COMPLETE INFORMATION

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

Program Number as a Resource

Program Number as a Resource is an option which can be specified on the $INFO PRGRES control statement in the Startup deck if the appropriate KEY module is present in the system.

Without this option, a slave job is assigned a program number when it is scheduled and is ready to be passed to the Peripheral Allocator for its initial allocation pass. The program number assigned remains with the job through its entire life cycle, which includes the initial allocation pass, a possible stay in limbo or in sieve, one or more waits in activity allocation, and then the various phases of execution.

With this option enabled, the program number is considered a resource and is allocated by the Peripheral Allocator. Following completion of the activity, the program number is deallocated by the Core Allocator thus allowing some other job that is about to begin an activity to use this program number.

The program number will be the first resource allocated to an activity. This must be done because FMS protected files use the program number as an index to various FMS internal tables. This provides program numbers for activities that are "waiting media" (which can be a relatively long wait) but that are necessary to preclude incurring the high overhead of having to deallocate and then reallocate FMS files for a new program number.

Exceptions to this are multi-activity jobs that have DAC remotes connected to them at the end of an activity. Such jobs keep the same program number for the next activity.

The aforementioned changes in the way program numbers are allocated have resulted in the fact that jobs in various stages of peripheral allocation are no longer visible through the hard core program number indexed table, but are found only in tables internal to the Peripheral Allocator. Even when the Program Number as a Resource option is not enabled, the true status of a job may sometimes be found only within the allocator.

The operator verbs LSTAL, LSTRT, and LSTPR, which report information relating to jobs in allocation, are processed under the Peripheral Allocator. Some of the functionality of the URGC and STATS verbs are performed by the Peripheral Allocator. The RWL verb is issued by SNUMB only.

TERMINATION

Three types of termination occur when running a program under GCOS.

- o Normal Termination
- o Abnormal Termination
- o Program Disaster

The termination procedure varies depending upon the type of termination.

Normal Termination

Normal activity occurs when a MME GEFINI is executed by a user program. If an activity terminates normally, the user will not get a dump, wrapup, or an abort subactivity, regardless of whether the user slave program prefix contains a wrapup address or whether the user input deck contains a $ ABORT card. An accounting record is sent to the Statistical Collection File (SCF), the execution report is generated, peripheral files are released, and memory is released.

If the activity is the last activity of the job (delimited by a $ ENDJOB card), a *EOJ message is issued on the console and the job is released for printer or card punch output. If this is not the last activity, subsequent activities are scheduled for allocation.

Abnormal Termination

Abnormal termination may be caused by (1) GCOS detected errors, (2) operator ABORT requests from the console, (3) hardware-generated faults, or (4) slave program requests (MME GEBORT). Any of these cause an immediate interruption of the activity. All the I/O requests for which the connect has not been issued are lost, however those I/O requests for which the connect has been issued are allowed to terminate normally.

The location of the fault or error and the abort reason code are placed in word 11 (decimal) of the slave program prefix. A *ABT message is output on the system console. If bit 0 of the program switch word is set, a dump is produced. If word 23 (decimal) of the slave program prefix contains a wrapup address, word 23 is cleared and the user is given control at that address. (This word is normally initialized by the .SETU. routine, if an Execute activity.) If no wrapup address is found, bit 12 of the program switch word is examined. If bit 12 is set, a Utility subactivity is executed. Upon return from this subactivity, the SCF and execution records are produced and peripherals and memory are released.

If termination was initiated by an ABORT request from the console, the job is terminated as though a $ ENDJOB followed. If termination was not initiated by an ABORT request and a $ ENDJOB is not the activity delimiter, the job is returned, with an abort indicator set, for subsequent activity allocation. In this case, only subsequent language processing activities (e.g., GMAP, FORTRAN) will be scheduled for allocation until a $ BREAK card is encountered, at which time normal activity scheduling is resumed. The conditional control cards, $ BREAK, $ GOTO, $ IF, $ SET, and $ WHEN, along with the $ Label cards may be used to explicitly control conditional activity execution.

In the case of a MME GEBORT, the program supplies only the reason code, and GCOS provides the rest. The instruction counter (IC) and reason code are inserted into word 11 (decimal) of the slave program prefix, as shown below.

Bits	0-17	IC
	18-23	02
	24-35	Reason code

If the job aborted, an entry will be made in the abort queue. This entry is then processed.

In order to prevent slave mode programs from abnormally terminating due to transient problems, a slave mode program abort recovery is provided through the JROLL option specified on the user's activity-defining control card. The JROLL option provides a rollback to a previous checkpoint where processing then proceeds. The use of this option is described in detail in the <u>Program Recovery/Restart</u> manual.

Abnormal Termination During Allocation

When the first pass through the job control file is completed, a test is made to determine if any errors were found during allocation. If so, the job is deleted, and a *DLT sssss (reason) message is sent to the console. Explicit reasons (see Appendix A) for deletion are printed on the execution report.

When a job is returned for allocation after execution of at least one activity and no subsequent activities are executable (because of fatal compile errors or activity abortion, for example), a *END sssss message is sent to the console.

If, because of hardware malfunction or unusual deck setup, a fault occurs during allocation, a *TILT, *DLT sssss message is sent to the console, a dump of the Peripheral Allocator is produced (on printer PR2, if dedicated, or on the SCF if PR2 is not dedicated), and the job is deleted from the system.

Dump, Wrapup, And Abort Subactivity

If an activity is terminated abnormally (either through a MME GEBORT or an abort) any, all, or none of the following may occur (user's option).

- o Dump (only if requested via a control card specifying the DUMP option or via MME GESETS).
- o Wrapup (which may also terminate normally or abnormally).
- o Abort Subactivity (which may terminate normally or abnormally).

The console log shows the first abort code; the execution report shows the codes in the order in which they occurred.

DUMP

A slave dump results only if the user has explicitly requested a dump by either:

1. Specifying the DUMP option on the control card which defines the activity.
2. Setting (via MME GESETS or $ SET) bit 0 of the program switch word to 1.

When the no dump option (NDUMP) is executed following an abort, a panel dump of the registers, the upper SSA, and the slave program prefix is produced.

WRAPUP

Following the possible dump, an optional wrapup is allowed. This occurs if a legal wrapup address is specified by the user in the upper half of word 23 (decimal) of the slave program prefix. Wrapup may terminate normally or it may undergo abnormal termination. If an abnormal termination occurs during wrapup, a dump does not occur. If a dump is desired, it must be done during wrapup via a MME GESNAP. The user is allowed the balance of the time specified (or implied) on the $ LIMITS card plus 40.96 seconds for wrapup.

In addition to supplying a legal wrapup address in the slave program prefix, the user must also supply a wrapup routine if it is not supplied by the software. In wrapup, the address (bits 0-17) of word 23 is checked for validity. If the address is out-of-bounds, no wrapup will be done.

ABORT SUBACTIVITY

An Abort subactivity may be specified by the user via a $ ABORT control card in his input deck. The subactivity is initiated if abnormal termination occurs and at least 8K of memory is available for the Utility program. If at least 8K is not available, an M4 abort occurs (see Appendix A). Again, if the abort subactivity undergoes abnormal termination, a wrapup may result. A dump does not occur.

TIME LIMITS FOR DUMP, WRAPUP, AND ABORT SUBACTIVITY

The maximum time allowed for any or all of the above (including any associated dumps in the case of wrapup and abort subactivity) is the time specified or implied on the $ LIMITS card plus 40.96 seconds for each wrapup and/or abort subactivity.

Program Disaster

If during the activity termination process, a fault or GCOS-detected error occurs, a *PROG nn DISASTER STATUS S#sssss message is issued to the console and the termination is stopped. A dump is then:

1. Printed on a dedicated printer if one is available.
2. If not, the dump goes to the SCF.
3. Termination is stopped with no files released and the normal termination procedure not completed.

If this type fault occurs, it may eventually result in a system failure.

SECTION VII

ACCOUNTING INFORMATION

The accounting information for an activity is produced during termination of that activity. It consists of information supplied in the following form:

- Execution Report
- Statistical Collection File

The lines on the execution report are described in this section. The binary accounting cards are described in the System Tables reference manual.

EXECUTION REPORT

The format of fields on the execution report is given in Figure 7-1. Since the actual content of the execution report can vary depending on the type of program, the execution report pictured in Figure 7-1 is only to illustrate the locations of the various executive report fields.

```
$$    ABC(1) ENTERED 10(2)   AT 12.654(3) FROM CD RDR(4) 0-06-00(5)

         (6)
0001  $      SNUMB   ABC,62
0002  $      IDENT   VXE00,JENNINGS P H,73180,K62
0003 A$      GMAP    NDECK
0004  $      LIMITS  10,32K
0005  $      UPDATE  LIST
0014 A$      EXECUTE
0015  $      PRIVITY
0016  $      LIMITS  999,2K
0017  $      ENDJOB
  TOTAL CARD COUNT THIS JOB = 000118(7)

 * ACTY -  01(8) $ CARD #0003  GMAP(9)  12/16/73(10)  SW=211400000000(11)

 * NORMAL TERMINATION     AT 002406(12) I=5020(13) SW=211000000000(14)

     START 12.659(15)    LINES     709(18)    PROC  0.0109(20)    I/O    0.003(22)   IU 62(24)  MEMORY   32K(26)

     STOP  12.672(16)    LIMIT  10000(19)    LIMIT 0.1000(21)    LIMIT         (23)   CU 62(25)  M*T     1626(27)

     SWAP    0.000(17)

     LAPSE   0.013(28) FC(29) D(30) TYPE(31)  BUSY(32)  IP/AT(33)  FP/RT(34)  IS/#C(35) MS/#E(36)  ADDRESS(37) T#/PK#(38)

                     G* R D191 *     375       0         8         8     8       0-03-02  #26370

                     A* R D191 *      38       0         2         2     2       0-03-02

                     P*  SYOUT

                     K*  SYOUT

                     C*  SYOUT

                     *1 R D191      7253       0         0         8    10       0-03-04  #11840

                     B* S D191 X      69       0         2         4     4
                                                                                 0-03-05  G481   ONL

      ⎧COMDK 1092  CARDS
  (39)⎨RC-77   87  LINES
      ⎪LIST  5695  LINES
      ⎩DECK   239  CARDS

     **** FILE 14 IS IN ABORT¹ (40)

     EXECUTE ACTIVITY DELETED¹ (41)
```

¹This line is not part of this activity, it is shown here to illustrate how it would appear on a printout.

Figure 7-1. Sample Format of Execution Report Fields

Definition of Fields in Execution Report

Index	Definition
1	SNUMB
2	System Identification--contains the six alphanumeric characters entered by the programmer on the $ SYID Startup control card
3	Time of day (in hours and thousandths of an hour)
4	Input source (RDR, IMCV, REMOTE,)
5	IOM number, channel (PUB) number, and device number
6	Control cards used in the program. Each activity definition card is marked by an A immediately preceding the $ symbol. If the program contains an object deck, both the $ OBJECT and $ DKEND cards will appear in the control cards list.
7	Total card count for this job
8	Activity number within job
9	Activity type (GMAP, BMC, etc.). (See Special Activity Output paragraph in this section.) The $ CARD field indicates the location at which the activity started.
10	Month/Day/Year
11	Status of program switch word at the beginning of activity
12	Location at activity termination
13	Contents of processor indicator register bits 18-29 at termination
14	Status of the program switch word at the end of the activity
15	Time activity started (in hours and thousandths of an hour)
16	Time activity stopped (in hours and thousandths of an hour)
17	Total time activity was swapped out of memory
18	Number of lines of output for this activity
19	Limit on number of lines of output for this activity
20	Processor time in hours and ten-thousandths of an hour
21	Limit on processor time for this activity (in hours and ten-thousandths of an hour)
22	I/O time for this activity (in hours and thousandths of an hour)
23	Limit on I/O time for this activity (in hours and thousandths of an hour). If this limit is not specified on the $ LIMITS control card, the I/O time is not limited and this field is blank on the execution report.
24	Initial urgency
25	Current urgency

26	Memory allocated to this activity (not including SSA)
27	Product of memory and time (in thousand word seconds)
28	Elapsed time (STOP -START) (in hours and thousandths of an hour)
29	File Code (FC)
30	Disposition Code: R Release S - Save D - Dismount C - Continue
31[1]	Media type according to the following list. Symbols *, P, S or X to the right of the device type indicates a System Input-created data file (*), a cataloged permanent file (P), a secondary file (S), or a removable file (X). The size is given in multiples of llinks (320 words).

Media Type	Device
DSU270	DSS270 Disk Storage Subsystem - Fixed
DSS167	DSS167 Disk Storage Subsystem - Removable
DSS170	DSS170 Disk Storage Subsystem - Removable
DSS180	DSS180 Disk Storage Subsystem - Removable
DSS181	DSS181 Disk Storage Subsystem - Removable
DSS190	DSS190 Disk Storage Subsystem - Removable
DSS191	DSS191 Disk Storage Subsystem - Removable
MSU310	MSU0310 Mass Storage Subsystem - Removable
MSU400	MSU0400/MSU0402 Mass Storage Subsystem - Removable
MSU450	MSU0450 Mass Storage Subsystem - Removable
MSU500	MSU0500 Mass Storage Subsystem - Fixed
MSU501	MSU0501 Mass Storage Subsystem - Fixed
BSS	Bulk Store Subsystem
CSS	DATANET 30/305 Front-End Network Processor
TAPE	Magnetic Tape Subsystem
TAPE-7	Seven-Track Magnetic Tape Subsystem
TAPE-9	Nine-Track Magnetic Tape Subsystem
TAP27	Seven-Track Magnetic Tape Subsystem in 2000/60 tape interchange mode
TAP29	Nine-Track Magnetic Tape Subsystem in 2000/60 tape interchange mode
READER	Card Reader
PUNCH	Card Punch
RD/PUN	CCU0401 Reader/Punch
PRINT	Printer
PPT	Paper Tape Subsystem
TYPE	Console Typewriter
CONSOL	System Control Center or CSU6001/6002/6601 Console
SYOUT	SYSOUT Function of GCOS (Device Independent)
S2P	Series 2000 Processor

[1] Media type P is the type of device on which the catalog resides, not the device that contains the file.

This entry identifies the type of device that contains the data file.

32 Amount of time the channel was busy. If _ 999999, time is in milliseconds. If 999999, time is shown in hours and decimal-fractions of an hour (e.g., 1.2345). A P following an entry in this column indicates a purged file.

33 For a mass storage device this field specifies the Initial Point (initial relative llink position) of the access. If the device is a unit record device, this field defines the Allocation Time (AT) for the device. For a magnetic tape subsystem, this field will be blank or will contain the channel address of an alternate magnetic tape unit.

34 For mass storage devices, this field (FP) describes the final position of the file in 320-word blocks from the start of the access, if available. If not available, this field contains NA (Not Available).

For magnetic tape devices, this field (FP) describes the final file position in the form r/ff, where ff is the number of files passed and r is the number of physical records passed. If the form is r ff, ff is the number of files passed, and r is the number of physical records backspaced in front of the last end-of-file.

For unit record devices, this field (RT) defines the Release Time.

35 For a mass storage device, this field (IS) contains the initial size of the file in llinks. For a unit record device or magnetic tape unit, this field (#C) contains the number of connects for the device.

36 For a mass storage device, this field (MS) contains the final size in llinks for the file. For a unit record device or magnetic tape unit, this field (#E) contains the number of errors in the access.

37 Device address (of device containing data file) in the form

IOM number - channel number - device number

38 This field (T#/PK#) contains the tape reel number (T#) or pack number (PK#).

For a magnetic tape unit, this field contains the File Serial Number (FSN) of the tape. When the FSN field of a tape reel matches the FSN field of the $ TAPE card (or the tape was a scratch reel) a # symbol precedes the FSN reported here. If the FSN's do not match at the start of an activity, a symbol precedes the FSN reported. For removable mass storage, this field contains the pack number.

For a removal disk pack (X following pack field), this field, on the line below the device definition line, contains the pack number followed by a field indicating whether the disk pack was online (ONL) or had to be mounted (MNT).

39	Lines printed (LIST) with SYSOUT report code 74 (octal) Compressed deck cards punched (COMDK) Object cards punched (DECK) Indicates a second report with a code of XX (XX not 74) has been requested.
40	One of the following messages may appear in this space: 1. **** FILE xx IS IN ABORT**** - An I-D-S file, number xx, is placed in abort status due to program abort. 2. **** FILE xx IS DEFECTIVE **** - Part of file, number xx, could not be read because of defective space.
41	Name of the activity deleted (if the activity was deleted)

SPECIAL ACTIVITY OUTPUT

GCOS service functions have the ability to write data on the execution report for the program containing that service function. The most common service functions that write data on the execution report are the label processing function of File and Record Control, and the printer button interface and $ MULTI card functions of the Bulk Media Conversion (BMC) program. The data is printed on the execution report between the line describing the activity and the line describing the activity termination.

BMC Activity Output

A BMC (CONVER) activity can output the following on the execution report for the program containing the activity:

- o Record of printer push-button operations on the printer dedicated to the BMC job.
- o Record of file manipulations caused by a $ MULTI card in the program.
- o Other messages as described in the _Bulk Media Conversion_ manual.

The printer control operation report may include any of the following messages:

```
FORWARD SPACE TO TOP OF PAGE
INVALID LINE
OPERATOR RESTARTED THIS REPORT
OPERATOR KILLED THIS REPORT
BACKSPACE TO TOP OF PAGE
BACKSPACE
```

For each entry on the $ MULTI card, BMC indicates the number of records input from a file or the number of records with a specified report code. These entries take the following form:

```
FILE#nnnnnn RPT CODE cc xxxxxx IGN yyyyyy

                    or

FILE#nnnnnn TOT xxxxxx INPUT

Where:  nnnnnn - File number
            cc - Report code (File and Record Control
                 report code)
        xxxxxx - Number of records used as input
        yyyyyy - Number of records ignored
```

The following is an example of this section of the execution report for a BMC activity.

```
  BMC- ASSEMBLED  070872
$       MULTI   1-1(21)/2(22)/3
NON-STANDARD EOF ENCOUNTERED; FILE MARK 16
FILE #000001  RPT  CODE   21  000100  IGN 000202
NON-STANDARD EOF ENCOUNTERED; FILE MARK 16
FILE #000002 RPT  CODE    22  000100  IGN 000202
NON-STANDARD EOF ENCOUNTERED; FILE MARK 16
FORWARD SPACE 1 LINE
FORWARD SPACE TO TOP OF PAGE
PREVIOUS LINE IN ERROR
BACKSPACE TO TOP OF PAGE
REPORT RESTARTED
REPORT TERMINATED
FILE #000003  TOT 000172 INPUT
 INPUT COUNT      000776      OUTPUT COUNT      000372
BLOCKS SKIPPED    000000      IGNORE COUNT      000404
```

File And Record Control Execution Report Output

The tape label processing function of the File and Record Control program prints a report of the label processing on the execution report. This data is the result of processing a multireel file or processing the first file of a multifile reel.

The messages are printed on the execution report between the activity definition line and the termination definition line. The general formats of the messages are as follows:

1. To describe the output reel the operator started with.

```
OPERATOR STARTED WITH #xxxxx FOR FILE CODE fc
                REEL SEQ #rrr FILE SEQ #sss OUTPUT

Where:  xxxxx - Tape reel number from output tape
           fc - File code defined on the $ TAPE card
          rrr - Reel sequence number of multireel file
          sss - Sequence number of file that processing started with
```

2. To continue a multireel file.

```
OPERATOR CONTINUED WITH #xxxxx FOR FILE CODE fc
                REEL SEQ #rrr FILE SEQ #sss
```

Where the variables are as defined for item 1.

3. To warn the programmer that the operator caused reel number xxxxx to be placed in EOR label of reel with sequence number rrr-1, but continued with reel number zzzzz on reel with sequence number rrr.

```
*****WARNING REEL #xxxxx FOR FILE CODE fc
                SEQ #rrr OUTPUT ERRONEOUS OPERATOR MOUNTED #zzzzz
```

```
Where:  xxxxx - Reel serial number in error
           fc - File code
          rrr - Reel sequence number
        zzzzz - Correct reel serial number
```

4. To provide the programmer with the reel serial number that was placed in word 14 of EOR label of reel with sequence #rrr-1.

```
OPERATOR ENTERED #xxxxx FOR FILE CODE fc
                    SEQ #rrr OUTPUT
```

```
Where:  xxxxx - Reel serial number of next reel
           fc - File code
          rrr - Reel sequence number
```

SUMMARY EDIT PROGRAM

The Summary Edit Program is a slave program that edits and outputs the information that has been accumulated on the GCOS Statistical Collection File. The output is collected on SYSOUT.

The program options include:

- o Printing all records and a summary report.
- o Printing only the summary report.
- o Printing only selected record types (1-9, 12, and 14-18) and a summary report. Type 8 records are printed out with Type 1. Type 9 and 12 records are always printed out.

Programming details and a sample of Summary Edit Program output with an explanation of each field appear in the Summary Edit Program reference manual.

SECTION VIII

SYSTEM STORAGE ENHANCEMENTS

In the basic system configuration, GCOS operates with both high speed main memory and extensive mass storage of all types. System performance can be further enhanced by adding options which expand the standard data storage capabilities of the system. These options are

- o Cache Memory *
- o Extended Memory

*

CACHE MEMORY

GCOS provides two cache memory options for increased program execution speed. The cache memory options give the processor approximately 100-nanosecond access to frequently used data and instructions. The cache memory is either a 2048- or 8196-word, high-speed buffer in the processor which reduces memory access time by making frequently used data and instructions immediately available to the processor. This can improve processor execution rates by 20 - 40 percent.

Individual processor identification bits define the presence and size of cache memory (2K or 8K). Software cache clearing is ignored by the hardware when hardware cache clearing is installed. If any processor has an 8K cache, hardware cache clearing is required in all SCUs. Multi-processor configurations may include 2K, 8K, or no cache memory at all. The cache printout on a master mode dump now includes the 8K cache buffer.

Cache memory design is based on extensive analysis and simulation studies, which reveal that generally a software operation using an instruction from a given memory location will, within a few hundred procedure steps, access a nearby location. For example, a procedural instruction string usually resides in a number of adjacent memory locations, and a procedure's operands are contained in tables so that they also reside in adjacent memory locations. To utilize this concept, cache memory is incorporated into the processor so that the processor can, on each memory access, bring in the requested information plus the information in the three adjacent memory locations. The extra information is stored in cache memory so that it is immediately available to the processor in succeeding operations.

Cache memory is divided into four-word blocks. Each block can be loaded with four contiguous words of main memory. Each block in cache memory is associated with a block of main memory locations via a directory in the cache memory control logic. This same control logic is used to determine where in cache memory to store a block of data from main memory.

The cache memory is basically transparent to the programmer. Cache memory has no program-dependent operating characteristics; therefore, it is neither necessary, nor desirable, to attempt to further improve processor performance by program modifications.

Normally, a processor with cache memory will operate with HEALS monitoring to provide a failsoft environment (see _System Operation Techniques_ reference manual for patch cards defining cache memory operations). In this mode, HEALS performs such functions as

- o Enabling the cache memory after system startup
- o Checking out cache memory during the enable
- o Monitoring cache memory errors and retrying instructions if required
- o Re-enabling cache memory if it has been disabled
- o Turning off a malfunctioning cache memory
- o Reporting cache memory errors on the system console

If HEALS monitoring is not desired, cache memory can be controlled by the Fault Processing module of GCOS. It is then enabled when GCOS first dispatches to a user program, and subsequently will be cleared just before the processor is given to a user program (enabled and cleared if cache memory was disabled).

Cache Memory Operation

Use of the INIT push-button on the console disables the cache memory, and the Startup program runs in a non-cache memory environment. Cache memory is enabled as the system begins operation. It can be enabled with or without monitoring by HEALS. The normal operation is with HEALS monitoring, but the user site can bypass this feature by applying a patch to the .MFALT module.

Operating under HEALS monitoring, cache memory is enabled when the HEALS Logging program performs its initialization. In this operation, the Logging program performs a brief checkout of the cache memory. Without HEALS monitoring, cache memory is enabled at the first dispatch to a user program.

Cache memory remains enabled during normal system operation. However, it is disabled when conditions occur that may result in a system abort. This may avert a system abort due to a fault within the cache memory, and saves the state of the cache memory control bits when a system abort occurs. These bits can be used in determining the cause of the abort.

If the system abort does not occur, cache memory is re-enabled and continues operation. If HEALS monitoring is used, cache memory is re-enabled when the HEALS Logging program makes its next periodic checks. If HEALS monitoring is not used, when the processor is dispatched to a program, cache memory will be enabled and cleared.

During normal operation, cache memory is cleared just before the processor is assigned to a program. It is also cleared when a Connect fault occurs in the processor. This protects against the possibility that the data in main memory may have been changed thus making it different from the corresponding locations in cache memory. (This facility also permits the I/O Supervisor to clear a cache memory on demand.)

Program Flow

Each processor data fetch is preceded by the generation of an absolute address of the data or instruction to be fetched. This absolute address is presented simultaneously to the memory port select logic and the cache memory directory logic. The memory port select logic prepares a memory cycle request, and the cache memory directory logic determines whether or not the requested data is in the cache memory.

If the data is present in cache memory, the data is given to the processor, and the main memory access is cancelled. If the data is not in cache memory, the memory access is performed and the data from two main memory locations is brought into the processor, where it is made available to the processing logic and entered into the cache memory at the same time. During the processing cycle, the next pair of memory locations is brought into the processor and entered into cache memory to complete the four-word cache memory block.

When the processor executes a Store instruction, one of two paths is taken. If the contents of the specified memory location is represented in the cache memory, both the main memory and cache memory are updated. If the specified memory location is not represented in cache memory, only the main memory location is updated.

NOTE: A Store to a memory location does not cause that location and the three adjacent locations to be brought into cache memory.

Use of the clear instructions (SZNC, LDAC, and LDQC) in a slave job will clear cache memory and thus cause the slave job to run slower.

Fault Conditions

If a system abort occurs, each processor's cache memory control bits are saved and the cache memory is disabled. The control bits can be used in the analysis of the abort.

Cache dump logic determines the state of the cache memory. The cache state word pointer is located at .MFALT entry point -1. The bits involved are as follows:

bit 18 = 1	cache 1 enabled (levels A and B)
bit 19 = 1	cache 2 enabled (levels C and D)
bit 20 = 1	operand enabled
bit 21 = 1	instruction enabled
bit 23 = 1	store aside enabled

The Read Switch Word command (RSW 2) preceded by LCPR set, 04 (set is a data word with bit 32 set to 1) loads the switches into the A register. Bit 27 set to 1 indicates that a 2K cache memory is configured; bit 20 set to 1 indicates that an 8K cache memory is configured. In a multiple processor system, it is possible that one processor may have an 8K cache memory and the next processor may have a 2K cache memory or no cache memory at all. After these checks are made, the "state line" is printed as follows:

PROCESSOR #n CACHE: =$\begin{Bmatrix}2\\8\end{Bmatrix}$ K CS1-E CS2-D INS-E S/A-E

where: E indicates enabled
D indicates disabled

Depending on the dump option exercised, a dump of the cache memory can be obtained. The Cache Memory Content Report appears in the dump output immediately preceding the trace table report. This content report contains a heading line, the cache memory directory content (512 entries), the cache memory content, and the closing line. Dump and trace functions are described in the System Startup reference manual.

HEALS Monitoring of Cache Memory

If HEALS is monitoring the operation of cache memory, it periodically checks for cache memory errors and to ensure that cache memory is still enabled. This check is made every five minutes or on a frequency specified by the site operators. This is done to detect directory parity errors, which are logged in the fault register. The HEALS logging program also checks for cache memory errors in each processor error record as it is processed.

The HEALS Logging program is also automatically enabled on the occurrence of a cache memory parity error, IA error on store, or parity error on block load.

HEALS operation with cache memory is described in the Honeywell Error Analysis and Logging System (HEALS) reference manual.

The HEALS Logging program monitors cache memory operation at five minute intervals. This frequency may be changed by using the console input verb

```
HEALS   DLYxx
```

Where: xx - User specified monitoring interval.

HEALS produces a variety of console messages that describe the cache memory operating conditions (refer to System Console Messages reference manual).

EXTENDED MEMORY

The extended memory[1] option, which permits access to up to 2M words of memory (M=1,048,576), provides greater depth to the multiprogramming capability of GCOS to increase system throughput. System overhead is reduced by the increased memory capacity because there is less swapping of programs.

The core extensions are made to a minimum 256K memory. The extension of memory and the maximum extension allowed are system model dependent and the Honeywell sales representative should be consulted for specific application.

Operation of the system using extended memory is transparent to user programs and privileged slave programs, which reference only their assigned memory area. The following description is intended only to provide a general overview of system operation in an extended memory environment.

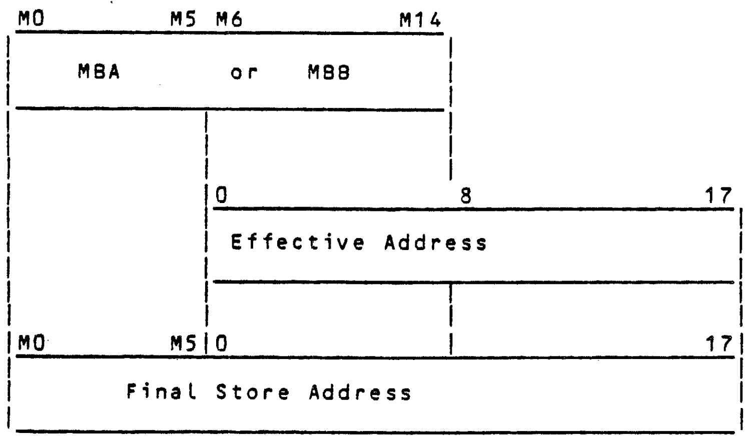
Extended Memory Addressing

The extended memory addressing option of the central processor extends to addressing capability of the processor to 24 bits. This is accomplished in slave mode operation by adding a six-bit Base Extension Register (BER) onto the high order end of the Base Address Register (BAR). In this extension of the BAR, only bits 4 and 5 of the BER are used at the present time. The bound field remains 256K, so that maximum quadrant size is 256K. However, the address extension permits the use of additional quadrants of memory.

In master mode operation, addresses are extended by the addition of one of a pair of 15-bit master mode BARs (MBA and MBB) to the effective address generated by the instruction to address memory.

[1] Memory capacities up to 2M total words is a standard product offering for Series 60 Level 66 systems. Memory capacities greater than 256K words is offered as an option on some Series 6000 systems. The extended memory operations described in this section apply to all systems utilizing more than 256K words of storage.

When the effective address of an instruction (including modification) executed in master mode is equal to or greater than 64K and a memory access is required, the contents of MBA or MBB are added to the effective address 0 modulo 512, thus producing a 24-bit address.



MBA is normally used for master mode Address extension. However, MBB may be specified by the programmer by the use of the new Load/Store instructions implemented for extended memory operation.

The use of MBA and MBB for addresses at or above 64K provides a 256K virtual memory in master mode consisting of the first 64K of memory storage (communications region and main memory modules) plus any other 192K determined by the value in MBA or MBB.

If the appropriate master mode BAR is set to the location of the slave program's lowest SSA origin, a virtual memory map exists, consisting of the first 64K of memory immediately followed by the slave program. This slave program always appears to have its origin at 64K plus the size of the SSAs assigned. For any given MBA setting, a 256K quadrant of memory storage is addressable.

Since the IOM cannot perform I/O across the 256K boundary, a slave program must be completely contained in one 256K quadrant of memory storage. When a program is put into execution by the Dispatcher, the proper value is loaded into MBA so that any GCOS module can access the 64K common memory area and the program in execution as in the past. The hardware automatically extends addresses to permit this operation.

As described, both instruction and operand fetches are based on the MBA register so that the true origin of a program is unimportant for accesses to a program for which the MBA is set (i.e., the program in execution). However, programmers preparing master mode programs must be aware that, as in other master mode operation, no boundary checks are made in using MBA and MBB. Care must be exercised in executing master mode instructions to protect other programs.

In standard GCOS, the .CRLAL table is used to load index register 5 or to absolutize an address, which is relative to the slave origin, for access to that location in master mode. Index Register 5 is automatically set by the Fault Processor module when entering master mode so the GCOS programmer may use it without regard to its actual content. This procedure is still required in Extended Memory GCOS, but the addressing process is carried one step farther by the extended memory addressing option, requiring GCOS to set the new registers to provide access to the desired memory address. This is done by using a pair of tables, .CRMBA and .CRMBB, which are used to contain the proper values to be loaded into MBA, MBB, and BER.

The Base Extension Register (BER), is loaded in conjunction with the standard BAR when a processor is dispatched to a program. The BER contains the number of the 256K memory quadrant in which the program resides. Its value is contained in the bits 18-35 of the .CRMBA entry for each program and is established by .MPOP5 at initial slave load and upon unswap of the program.

The primary master base address register (MBA) is likewise set by the Dispatcher upon putting a program into execution. Its value is kept in the allocator module that modifies the BER value.

The alternate master base address register (MBB) is invoked only by use of the special instructions MLDA, MLDQ, MLDAQ, MSTA, MSTQ, and MSTAQ. This permits data fetches or stores to a 192K memory storage window, which may be different from the one defined by MBA. This is necessary when a program such as .MPOPM or VIDEO must access memory space in a different program, which may or may not be in the same 256K memory quadrant. When this is necessary, the executing program will use MBA for instruction fetches, so MBB must be loaded from the .CRMBA entry for the other program, and one of the special instructions is then used to fetch/store data in that program. .CRLAL must still be used to absolutize slave addresses, however. The contents of MBB are saved in the .CRMBB table upon any program interruption and restored from there upon return or redispatch to that program. Thus, if MBB is to be altered in courtesy call, it must be saved, used, and restored in inhibited coding to prevent destruction of the value saved in .CRMBB at main level.

Example

The following example assumes three quadrants of memory storage. It is important to note that the values contained in .CRLAL (and consequently in index register 5) no longer reflect the true origin of the slave program, but only the virtual origin. Thus any function dealing with true program origins, such as system console messages, must include the MBA value in its calculations.

Program Number	Program Origin(8)	Program Size(8)	SSA Size(8)	MBA	BER	.SALIM (BAR)	.CRLAL (X5)
1	102000	010000	2000	0	0	102010	102000
10	314000	304000	4000	00110	0	314304	204000
11	1504000	274000	4000	01300	1	504274	204000
14	622000	156000	2000	00420	0	622156	202000
16	2004000	574000	4000	01600	2	004574	204000
20	160000	32000	2000	77756	0	160032	202000

The address of the instruction LDA .STATE,5 is handled as follows:

Program Number	Effective Address	Final Address
1	101040	101040
10	203040	313040
11	203040	1503040
14	201040	621040
16	203040	2003040
20	201040	157040

When it is necessary to access the .STATE word of a program other than the one for which MBA and index registers 5 and 6 have been set, the technique is as follows (when program number is in Xi):

```
LDXj      .CRLAL,i      get LAL value

LMBB      .CRMBA,*i     get MBA value

MLDA      .STATE,j      use MBB to locate .STATE
```

Use of such coding still requires proper gating procedures, and in fact may be more critical to lack of gates since no direct compare may be done using MBB. Hence, additional instructions may be required to accomplish a given task.

File Management Supervisor in Extended Memory Environment

In an extended memory system, the File Management Supervisor (FMS) must reside in the first 64K of memory. This permits its tables and routines to be referenced directly by the FMS Slave Service Area modules running under other program numbers.

FMS also has the capability to use the special instructions associated with extended memory to access information in other programs.

SECTION IX

GCOS UTILITY FUNCTIONS

VIDEO

The Visual Information Display for Efficient Operation (VIDEO) program is used to display system operating statistics on a VIP terminal (keyboard/display device) or on a System Control Center (SCC) configured in a VIDEO mode (identified as TY5 in the $CONFIG section of the Startup deck). VIDEO runs as a direct program access (DAC) privileged slave program under GCOS.

VIDEO Operation

The VIDEO program is lowloaded for internal memory management. Since only one Slave Service Area (SSA) is required, the $ LIMITS card for this program should specify 2K for SYSOUT to ensure 1K for the SSA.

KEYBOARD/DISPLAY TERMINAL

Initialize VIDEO for a keyboard/display terminal as follows:

1. Load the program deck. The SNUMB of the program is immaterial.

2. When VIDEO is in execution, issue the following connect from a keyboard/display terminal:

   ```
   $*$idbuseridbpassword,nn,DAC,VIDEO
   ```

   ```
   Where: id - Station identification of terminal
    password - Password (site parameter) for access to the system.
          nn - Screen size is 22 lines
   ```

   DAC is used only if connect is through a DATANET 30 Front-End Network Processor, operating under the G30 program (Series 6000 only). The connect name of this program is always VIDEO, no matter what SNUMB is on the job.

3. VIDEO responds with a display of system information (normal response) or with the following message (abnormal response):

   ```
   *VIDEO: CONNECT REJECTED
   ```

In this type initialization, the first terminal to connect to VIDEO must be a keyboard/display terminal. Other terminal types can then attempt to access VIDEO and will be treated as mailbox inquiries (see paragraph on options).

During operation, VIDEO maintains an outstanding remote inquiry to allow additional keyboard/display terminals to request access, using the connect sequence described above. When a line is accepted, VIDEO responds with the following message:

```
* VIDEO: CONNECT ACCEPTED
```

The next time VIDEO data is updated, a screen of data is issued to the newly connected terminal. A maximum of four keyboard/display terminals may be connected to VIDEO at the same time. If more than four terminals attempt access, a *VIDEO LINE TABLE FULL message is sent to the requesting terminal, followed by a line disconnect.

SYSTEM CONTROL CENTER

A System Control Center (SCC) may be configured at system startup for operation as a VIDEO terminal. This configuration is accomplished by defining a SCC as TY5 in the $CONFIG section of the Startup deck.

When the VIDEO program is initialized, it searches for a SCC device configured with the name TY5. When this SCC is found, VIDEO issues a screen of data to the SCC.

VIDEO Display

The VIDEO display is formatted as follows:

1. The first line contains a system status label followed by the date and time (time in hours and thousandths of an hour) of the last update. The date is reset automatically if VIDEO is in execution across midnight.

2. The next line contains the labels for the jobs waiting, jobs executing, and System Scheduler stacks. A total number of jobs appears at the top of the jobs waiting and jobs executing stacks.

3. In the lower center of the screen is the SYSOUT stack, with an entry at the top to show the total number of jobs remaining in the GEOT SNUML table to be processed.

4. At right center is the peripheral allocation stack.

5. At the lower right is a system summary.

6. At the lower left is user identification stack (TSS USERS) for the Time Sharing System.

When there are more jobs in a stack than can be listed in one display, the last entry in the stack will be **MORE**, and the next update will continue where the incomplete list left off. The counts at the top of the waiting, executing and SYSOUT stacks are true total counts, even when the full stack cannot be listed in one display.

JOBS NOT DISPLAYED DURING VIDEO UPDATE

When the Peripheral Allocator is swapped out during VIDEO's update pass, information pertaining to jobs awaiting allocation is not displayed. Also, because the System Configuration Tables of saved peripherals show a 2 as the program number between activities, the output of VIDEO and such verbs as TYPFG does not specify the exact SNUMB that owns such peripherals when the owner is in the activity allocation stage.

JOBS WAITING STACK

This stack lists, in program number order, the jobs waiting for execution. In this list, each job is identified by the sequence number (SNUMB) and current activity number. To the right of each SNUMB, a three-character status code indicates why the job is waiting. The codes are as follows:

RMT	Reading, Disk
ALC	Waiting for allocator
*PER	Waiting for peripherals
*COR	Waiting for memory storage
HLD	In HOLD status
*LIM	In limbo
MNT	Waiting for tape mount/ready
SIE	In SIEVE status
*OVD	Allocation overdue
RST	In restart
TRM	In termination

Statuses marked with asterisk (*) also have one of the following explanations to the left of the SNUMB.

1. If the job is in limbo, an octal number (#nn) for use in RUN requests appears at the left of the SNUMB.

2. If the job is waiting for memory, the number of 1024-word blocks needed (including SSAs) appears as a three-digit number at the left of the SNUMB.

If the links required exceed 999, the asterisk (*) will be used as a special descriptor.

TSS USERS STACK

This stack displays the user identifications of active Time Sharing System users and the corresponding station identifications. A maximum of thirty user identifications can be displayed (including stack roll). To display more or less user identifications, the following alter must be assembled with VIDEO:

```
$         ALTER        47,47
NTSMAX    EQU          nn
```

Where: nn - Number of user identifications to be displayed.

JOBS EXECUTING STACK

This list, ordered by program number (SNUMB), describes the jobs that are in execution or swapped out (other than system programs and VIDEO). Jobs that are swapped out are identified by an S to the left of the SNUMB. Jobs in termination are identified by a T to the left of the SNUMB.

SYSOUT STACK

This list describes all jobs waiting to be processed by SYSOUT. This stack is ordered according to the SNUML table. Two indicators on the left of the SNUMB define the disposition of the job:

1. An asterisk (*) in the second character position of SYSOUT entry indicates that printer output for that job is waiting.
2. An asterisk (*) in the third character position of the entry indicates that punched card output is waiting.

If either output is in process, a plus sign (+) appears as the indicator symbol.

If a job has remote output waiting, the remote station identification is displayed to the right of the job's SNUMB.

If more than one station identification exists, they are used in the order in which they are taken from the SYSOUT Report Writer's SNUML table. If output is being transmitted to a station, the SNUMB and station identification are joined by a hyphen. When the Report Writer is swapped out, this stack is empty.

SYSTEM SCHEDULER STACK

This list describes the status of the System Scheduler operation. This display is ordered as follows:

MAX/MIN values

Number of jobs in EXPRESS class

Number of jobs in HOLD class

Total jobs in all other classes

PERIPHERAL ALLOCATION STACK

This list contains the status of all unit record devices, tape peripherals, and removable disk packs named when VIDEO was initialized. The display consists of a three-character device name (represented by xxx) and one of the following definitions:

xxx:(blank)	Device free
xxx:SNUMB	Device allocated to job identified by SNUMB* (including system programs)
xxx:RLSED	Device released
xxx:DEDCT	Device free but dedicated
xxx:ACCNT	Device is system accounting collector

Jobs are listed in this stack in order by ascending IOM number, within the IOM by channel (PUB) number, and within the channel by device number. An exception to this is a pair of crossbarred channels. In this case, all devices are picked up from the primary channel, even if it has a higher number than the secondary channel.

SYSTEM SUMMARY SECTION

Each time VIDEO refreshes a screen of information, the system summary section of the display is updated. The entries in this stack are as follows:

CORE: xxx/yyy

Where: xxx - Number of 1024-word blocks of memory storage available for allocation

yyy - Largest single contiguous block available for allocation

JOBS: xxx/yyy

Where: xxx - Total number of batch jobs run since bootload or HISTRE

yyy - Total number of remote jobs run since bootload or HISTRE

TSS: xx/yy

Where: xx - Current number of Time Sharing System users

yy - Current size of Time Sharing System in 1024-word blocks

TSS: SWPD

Time Sharing System swapped out

TSS: OFF

Time Sharing System not in execution

TAPE: nn

Where: nn - Total number of tape handlers available for allocation

SCC Hard-Copy Output

On the System Control Center operating in the VIDEO mode, a hard copy of the display can be output on the SCC printer. To get this hard-copy output,

1. Switch the SCC printer to the channel being used for the VIDEO display.
2. Switch the printer to online.
3. When the display printout is completed, return the printer to offline.

VIDEO Options

KEYBOARD/DISPLAY TERMINAL

Options available for the VIDEO program are entered after forcing a Break at the keyboard/display terminal. The Break is forced by entering the following control record at the terminal.

```
$*$BRK
```

The VIDEO program responds by transmitting the screen image.

```
TIME:  nn SECS
LINE:  m
MAIL:  ***
MESS:  ***
MODE:  mod
OPTION?
```

Where: nn - Current update frequency of data in seconds (1_nn_63).

m - Current number of lines connected to LSTAL.

*** - Status of mailbox and all points bulletin, ON or OFF.

mod - Current mode indicator is *NRM (normal) or *CON (continuous)

In response to the OPTION? query, the operator should enter a device option. The option verb is the first four nonblank characters in the input stream. For verbs longer than four characters, only the first four characters are displayed. If the option entered is illegal, VIDEO responds WHAT? and waits for a valid option to be entered. A verb of all blanks is ignored. When the verb changes the state of an option, the initial assumption is noted.

SCC TERMINAL

On a SCC configured for VIDEO operations, the input request is initiated by pressing the KEYBOARD SELECT push-button to switch to the channel connected to TY5 and pressing the REQ key on the keyboard. VIDEO responds with

VIDEO?

The operator may then enter any of the VIDEO options.

The option verbs and their functions are as follows:

o VOCAB	Causes a list of all valid VIDEO options to be sent to the requesting terminal. Following this display, VIDEO requests another option.
o nn	Change update interval to value (in seconds) given as nn ($1 \leq nn \leq 63$). If value exceeds 63, 63 is used as the update interval.
o *NRM	Places VIDEO in continuous mode and VIDEO updates display according to the specified sampling interval. This verb is used to return VIDEO to normal mode from continuous mode. The normal mode uses the sampling interval in effect at the time continuous mode was initiated.
o *CON	Places VIDEO in continuous mode and VIDEO updates display after transmitting a screen image to the terminals. Time between updates depends mainly on the number of lines connected. This mode is cancelled by entering a new sample time or by entering *NRM.
o *EOJ	Terminates VIDEO as follows: (1) disconnects all lines, (2) issues ABORT request, and (3) terminates via MME GEFINI.
o ABORT	Terminates VIDEO via MME GEBORT. The allocator patch is replaced by wrapup.
o *NEW	Allows additional terminals to access VIDEO. This option is used only to cancel the NONEW option.
o NONEW	Prevents new terminal accessing VIDEO. A terminal attempting to access VIDEO receives the message *VIDEO LINE TABLE FULL, even if the table is not full.
o TIME ABCD	Indicates computer system unavailability as follows: 1. Sends following message to all lines. System UNAVAILABLE UNTIL time 2. Disconnects all lines. 3. Waits for new control line.

o LINES	List the line identifications as follows: CURRENT LINE ID'S:I1,I2,...,In VIDEO then requests another option.
o DISC ID	Disconnects the line identified by identification from VIDEO. If the specified line is not connected to VIDEO, the request is ignored.
o NDSP options	Prevents display of SNUMB. Options are: ADD SNUMB REM SNUMB CLR VIDEO places SNUMB's specified in this verb in a table and does not display them. The REM SNUMB option removes SNUMB from this table. The CLR option erases this table.
o MESS options	Stores and outputs a message to all connected keyboard/display terminals at a specified interval. Options are: LOAD OFF ON FREQ nn The LOAD option is used to load the message and VIDEO issues the following message to the requesting terminal: READY MAXIMUM SIZE IS 8 LINES The operator responds with a message up to eight lines long. VIDEO continues to update all other terminals while waiting for message input. When the message is input, it is transmitted every n (n initially 10) updates in place of the status information. The frequency can be changed using the FREQ nn option, with $1 \leq nn \leq 31$. If nn = 0, the request is ignored; if greater than 31, 31 is used. The OFF option stops message transmission, but does not disturb the contents of the message buffer. The ON option reactivates the message. If MESS ON is used and the buffer is empty, the request is ignored.

- MAIL options — Provides a message for teletypewriter terminals. The options are:

  LOAD
  OFF
  ON

  VIDEO stores and outputs a message to any teletypewriter terminal that attempts to access VIDEO. Following the message, the terminal is disconnected. When the LOAD option is used to load a message, VIDEO responds:

  ```
  READY MAXIMUM SIZE IS 6 LINES
  ```

  The operator enters a message up to six lines long. VIDEO continues updating the screen image while waiting for the message to be input. If the message buffer is empty on a connect request, VIDEO sends the terminal a message, followed by a disconnect.

  ```
  *VIDEO: MAILBOX EMPTY
  ```

  The OFF option prevents transmission of the message loaded via LOAD, but does not disturb the contents of the message buffer. The ON option is used to restore message transmission.

  In addition to the message sent to the terminal, VIDEO also sends a system summary which is identical to a VIDEO screen display. The STAT OFF option blocks this summary transmission. The STAT ON option restores the summary transmission.

- STAT OFF/ON — STAT OFF blocks the transmission of the system summary to the terminal (see MAIL options). STAT ON restores this transmission.

- DATE — VIDEO resets the display date from .CRDAT.

- TSSU ON/OFF — Displays Time Sharing System user identifications. The options are:

  ON
  OFF

  VIDEO displays the Time Sharing System user identifications unless TSSU OFF is entered as option. It resumes the display when the TSSU ON option is exercised.

- SWAP ON/OFF — SWAP ON causes VIDEO to set its urgency to 0 so that it can be swapped out. SWAP OFF is used to return from the SWAP ON condition.

- o MOVE — Permits an operator to move an interactive system console name to a SCC. A request entered at a system console to move an interactive system console function (TY1, TY2, TY3, or TY4) to TY5 is denied if TY5 is currently assigned to VIDEO. However, if the request is made to VIDEO, VIDEO relinquishes control of the SCC to accomplish the move. MOVE requests for interactive system console functions cause the following:

  1. Validation of the request.
  2. Association of the system name with a different device in the System Name Table.
  3. Removal of .MPOPM's GESPECed I/O queue (associated with that name) from the old SCT.
  4. Link of this I/O queue to the new device SCT.

Restoring VIDEO Display On SCC

When an interactive System Control Center (SCC) is taken out of service, an interactive console name may be moved to the SCC channel named TY5 (via a MOVE verb). If the VIDEO program is running on channel TY5, the program terminates itself unless there are VIP terminals connected to the program. If VIP terminals are connected, the VIDEO program continues to display system operating statistics on them.

If the VIDEO program is running on the VIP terminals, the display can be restored to the SCC (TY5). This is accomplished by entering the VIDEO verb on the interactive console. The VIDEO program checks bit 11 of @.CROPT. The VIDEO verb sets this bit to tell the program that TY5 is available, and the VIDEO display is restored to TY5.

If the VIDEO program terminated itself because no VIP terminals were connected, the VIDEO program must be re-initialized to restore the SCC (TY5) to the VIDEO mode of operation.

BMC SPINOFF

Bulk Media Conversion (BMC) $ CONVER activities, which convert large tape input files to printer or card punch output, may be spun off the main job and run as independent jobs. With the activity spun off, the remaining activities in the job are executed without waiting for the completion of the $ CONVER activity. This prevents the slowdown in execution of a job which contains a $ CONVER activity that might be held up waiting for an output device.

The spinoff function uses the MME GENEWS instruction as the spawning vehicle. This permits spinoff activities to be restarted after a system fault. It also allows an input tape for the activity to be deallocated and dismounted until the console operator is ready to execute the activity.

To initiate the activity, the console operator enters a RUN nn verb for the dismounted reel. The requirement for this verb gives the operator control of the number of active spinoff jobs in the system and of the sequence in which they are to be run. The tape dismount may prevent unnecessary commitment of tape resources while waiting for a printer or card punch.

The following is a sample console log for this type of spinoff operation:

```
               *SRT  SSSSS-01    11.931  GMAP       (040)

     *DMT  0-02-00  REEL  #23558 S# SSSSS-02 TO 4940S B-M-C

REMOVE OUTPUT SSSSS-04 FROM PRINTER # 0 09

               *EOJ  SSSSS-05   11.942

          *QUE EMPTY   11.942

     *GET P#10 S#4940S TAPE#23558 SPINOF-SSSSS

???RUN 10ƀ

     *MEDIA--S#4940S-01  MNT 0-02-00  #23558  SPINOF-SSSSS   11.947

               *SRT  4940S-01   11.949  CONVER  (080)

REMOVE OUTPUT 4940S-01 FROM PRINTER # 0 09

               *EOJ  4940S-01   11.953

          *QUE EMPTY   11.953

          *EXEUNT GEOT
```

In this example, SSSSS is the parent job and 4940S is the spawned job. The messages give the operator more information and control of the activity.

APPENDIX A

TERMINATE MESSAGES, ABORT AND DELETE REASON CODES

CONSOLE OUTPUT MESSAGES

ABORTS: When caused by hardware faults or GCOS detected errors, aborts are identified by a message text in the form:

```
*ABT sssss-aa  tt.ttt (abort reason)
```

The abort reason is a four-word message output on the console. The octal code identifying the message is contained in bits 24-35 of the Q-register.

When caused by errors other than those above (i.e., when not detected by GCOS), aborts are identified by a two-character code at cc in the message:

```
*ABT sssss-aa  tt.ttt *USER cc
```

DELETES: When, due to a fault during allocation that caused an unrecoverable condition, deletes are identified by the message:

```
*TILT, DLT S#sssss
```

1. A TILT message usually indicates that the system is going down.
2. If, however, a job is deleted during allocation for subsequent activities after having executed at least one activity, the following message appears:

   ```
   END sssss  tt.ttt ACTY DELETE
   ```

When due to fatal control card errors, deletes are identified by the message:

```
*DLT S#sssss-aa IMPROPER CONTROL CARDS
```

EXECUTION REPORT OUTPUT MESSAGES

ABORTS:	Abort message texts and/or reason codes (see discussion under Console Output) appear in place of the NORMAL TERMINATION portion of the termination message following the J* (control card) file listing.
DELETES:	Messages giving the reasons for deletes by the System Input program are imbedded in the J* file listing. All other GCOS delete messages appear following the J* listing.

Dump Analyzer Messages

AT	Abnormal termination caused by ABORT console verb.

Bulk Media Conversion Messages

B0	Partitioned record error.
B1	Improper use of variables on a $ OUTPUT or $ INPUT card. Check for spelling, use of MIXL on input, etc.
B2	Variables on the $ MULTI control card are incorrectly used.
B3	Improper request for transliteration. Either the media combination is in error or codes (IBMF, IBMC, IBMEL, IBMPL, or G-225) are improperly interpreted. Transliteration applies only to card reader, card punch, and magnetic tape units (only card reader and card punch for IBMEL or IBMPL); assigning it to any other device causes a B3 abort.
B4	Input or output device code is not acceptable to BMC. This message indicates that the device type for the IN or OT file was not acceptable as INPUT or OUTPUT, or that no IN or OT file was given to BMC.
B5	A partial header label has been encountered on magnetic tape input.
B6	User has not provided an Sxxxxx or Uxxxxx option (input only), and an error has occurred.
B7	The number (xxxxx) of acceptable input errors as specified in Uxxxxx or Sxxxxx has been exceeded.
B8	Input error prevents BMC from continuing.
B9	An entry made to an overlay is erroneous; probable hardware failure.

COBOL System Messages

C0	A requested input/output file is not contained in the file address table; an internal problem in the compiler. (Notify Honeywell field representative.)
C1	Address table (*3 File) is full. Normally, this problem may be corrected by increasing the size of the *3 File with a $ FILE card. The maximum is 106 random links.
C2	Fixed portion of the data name table is filled to the point that the largest record (01) contained on the overflow file cannot be read back into memory. Normally, this problem may be corrected by including a $ LIMITS card to extend memory limits, thus enabling extension of the data name table.
C3	Invalid internal list structure of the Data Division, possibly caused by invalid level structures for some data items which have been processed prior to this point.
C4	Substantive Stack build error in Report Writer (internal compiler problem).
C5	Invalid internal list structures of the Report Writer (internal compiler problem).
C6	Fixed portion of the Report Table is full. See <u>COBOL User's Guide</u> for ways in which changes can be made to the Report Section (report descriptions) of the source program so that abort is eliminated.
C7	Fixed and variable portion of the Report Table is full. See COBOL User's Guide for ways in which changes can be made to the Report Section (report descriptions) of the source program so that abort is eliminated.
C8	Invalid internal list structures of the Report Writer Analyzers (internal compiler problem).
C9	Report Writer Generator is unable to build its COBOL lists (internal language) using the Analyzer build routines (internal compiler problem).
CA	Invalid internal list structures of the Report Writer Generators (internal compiler problem).
CB	Invalid (not 17 or 23 octal) end-of-file mark has been encountered in a COBOL object program.
CC	Compiler has developed an internal syntax processing error (internal compiler problem).
CD	Copy module needs more memory. Adjust $ LIMITS card, if possible.
CE	Stack overflow in processing source syntax (internal compiler problem).
CF	Attempted RETURN statement without a SORT or MERGE in control.
CG	A COBOL object program has attempted one of the following: 1. The value zero exponentiated by the value zero. 2. The value zero exponentiated by a negative value. 3. A negative value exponentiated by a nonintegral value.

CH	A derail return is received when a MME GESYOT is executed to engage the back door file facility.
CI	An attempt to engage the back door file facility was made when the facility was not included in the system configuration.
CJ	Misspelled or missing PROCEDURE division card.
CL	Invalid LIST structures encountered by the internal language analyzers (internal COBOL problem).
CR	I/O incomplete for a random file in a COBOL object program.
CT	Attempt to load test monitor dummy link was unsuccessful (internal COBOL problem).

COBOL - Transaction Processing System Interface Messages

CX	Content of OUTPUT-SIZE field specifies a message larger than the message that can be contained in the sending field.
CZ	No station currently attempting to access the message.

I-D-S System Message

D2	I-D-S has aborted the program because the accounting buffer is too small for the I-D-S record to be journalized.

Dump Analyzer Messages

DR	Derail fault caused by dump analyzer.
DV	Divide check fault.
EF	Nonrecoverable error in data on dump tape.
FT	Fault tag modifier.

.MFALT Internal Fault Codes

F-1	Execute Fault Code generated by Software given to Processors not causing fault.
F0	MME
F1	Memory
F2	Fault Tag
F3	Command
F4	Derail
F5	Lockup

F6	Illegal Op Code
F7	Op Not Complete
F10	Overflow
F11	Divide Check
F12	Timer Runout
F13	Parity
F14	Not used
F15	System Abort Dump Request (FSB)
F16	GWAIT error gate closed more than 500 msec.

ABORT MESSAGES

File And Record Control Messages

G1	Blank tape rather than input header label.
G2	Error in Input Tape Header label on tape.
G3	Error in Output Tape Header label.
G4	Error in old label on output tape.
G5	End-of-tape while writing header label.
G6	Blank tape rather than input trailer label.
G7	Block count error in input tape trailer label.
G8	End-of-tape detected on multifile reel. (Note: A multifile reel cannot be continued on a second tape.)
G[	Wrong density specified for input tape header label.
GC	Unable to write trailer-label sequence.
GF	File and Record Control has aborted program for the cause indicated by its abort message. GF errors usually indicate a program error and the program should not be rerun.
GR	File and Record Control has aborted the program for the cause indicated by its abort message. GR errors indicate it may be possible to rerun the program successfully.
NT	An attempt has been made to access a terminal without having loaded the proper routine; .RTYP option entered as SYMREF on the $ USE control card.

NOTE: Index register 2 contains the location of the FCB of the file that caused the error.

GF and GR Abort Messages

```
ABORTED BY GEFRC ROUTINE rrrrrr CODE x FILE CODE cc
    ┌──────────────────────────┘       │          ↑
    │    ┌─────────────────────────────┘          │
    │    │                            L   FCB   FOR  | IS      IN  XR2
    ↓    ↓
```

BSREC	1	ILLEGAL REQUEST FOR THIS ROUTINE
	2	STATUS NOT TAPE ON LOAD POINT OR OK
BSTFM	1	ILLEGAL REQUEST FOR THIS ROUTINE
	2	REACHED LOAD POINT BUT MORE FILES TO SKIP
CLOSE	1	EOF STATUS ON OUTPUT FILE
	2	UNRECOVERABLE I/O ERROR,NO USER ROUTINE
	3	FILE TO BE CLOSED IS NOT IN CHAIN
	4	ILLEGAL STATUS FOR DISK OR DRUM
	5	UNRECOVERABLE I/O WRITING EOF MARK ON TAPE
	6	CHECK POINT RETURNED ERROR STATUS
CHECK	1	CHECK POINT RETURNED ERROR STATUS AFTER VAL SW
FORCE	1	ILLEGAL REQUEST FOR THIS ROUTINE
FSREC	1	FILE IS NOT PRESENT
	2	EOF ON DEVICE FROM PRIOR COMMAND
	3	IMPOSSIBLE RETURN FROM SYSTEM ROUTINE
	4	ILLEGAL FILE DEFINITION IN FCB
	5	UNRECOVERABLE I/O ERROR,NO USER ROUTINE
	6	ILLEGAL REQUEST FOR THIS ROUTINE
	7	I/O STATUS OTHER THAN BLANK TAPE ON READ
FSTFM	1	ILLEGAL REQUEST FOR THIS ROUTINE
GET	1	FILE DESIGNATED AS OUTPUT FILE
	2	ILLEGAL FILE DEFINITION IN FCB
	3	UNRECOVERABLE I/O ERROR, NO USER ROUTINE
	4	BLOCK SERIAL NUMBERS OR BLOCK LENGTH ERROR
	5	FIXED OR MIXED REC. SIZE FOR A FILE IS ZERO, OR VAR. REC. SIZE IS ZERO FOR TAPE FILE
GF200	1	EOF STATUS ON OUTPUT FILE
	2	UNRECOVERABLE I/O ERROR, NO USER ROUTINE
GF275	1	UNRECOVERABLE I/O WRITING EOF MARK ON TAPE
GF980	1	TRIED TO CREATE AN ILLEGAL I/O REQUEST
GXLIT	1	INCONSISTENT FILE DESCRIPTION
	2	ILLEGAL FILE CODE
	3	INVALID CARD OPTION
	4	TOO MANY TRANSLITERATED FILES
	5	I/O ERROR; NO RECOVERY ATTEMPTED
	6	ILLEGAL TRANSLITERATED DEVICE
	7	BLOCK SIZE TOO LARGE
	8	USER BUFFER TOO SMALL
	9	REQUESTED PARAMETER NOT IMPLEMENTED
OPEN	1	ILLEGAL DEVICE CODE FOR GEFRC
	2	FILE IS LOCKED
	3	DEVICE IS PRINTER OR PUNCH,NOT OUTPUT FILE
	4	ILLEGAL DISK OR DRUM FORMAT
	5	LINKED FILE BUT NOT VARIABLE RECORD TYPE
	6	ILLEGAL FORMAT FOR SYSOUT FILE
	7	FILE DESIGNATED AS REQUIRED IS NOT PRESENT
	8	TWO FILE DESIGNATORS POINTING TO SAME FILE
POST	1	FILE NOT ACCESSIBLE
	2	UNRECOVERABLE I/O ERROR, NO USER ROUTINE
	3	UNRECOVERABLE I/O ERROR, MULTI-POST
PUT	1	EOF STATUS ON OUTPUT FILE
	2	UNRECOVERABLE I/O ERROR,NO USER ROUTINE
	3	ILLEGAL FILE DEFINITION IN FCB
	4	CURRENT LOGICAL RECORD LARGER THAN BUFFER
PUTSZ	1	ILLEGAL REQUEST FOR THIS ROUTINE
	2	NEW SIZE LARGER THAN OLD RECORD SIZE

```
ABORTED BY GEFRC ROUTINE rrrrrr CODE x FILE CODE cc

                                          L  FCB   FOR   IS  IN  XR2

RDREC   1  BINARY CARD IS NOT A COMDEK CARD
        2  THE COMDEK CARDS ARE OUT OF SEQUENCE
        3  DATA ERROR IN DECOMPRESSING COMDEK CARD
READ    1  ILLEGAL FILE DEFINITION IN FCB
REMOT   1  REMOTE TERMINAL OUTPUT FILE HAS NO BUFFER
REWND   1  ILLEGAL REQUEST FOR THIS ROUTINE
USERR   1  ROUTINE CALLING ERROR WAS USED BY SAME
WAIT    1  UNRECOVERABLE I/O ERROR, NO USER ROUTINE
WEF     1  EOF ON OUTPUT FILE
        2  UNRECOVERABLE I/O ERROR, NO USER ROUTINE
        3  ILLEGAL REQUEST FOR THIS ROUTINE
        4  ILLEGAL STATUS FOR DISK, DRUM OR TAPE
WRITE   1  FILE IS NOT PRESENT
        2  EOF STATUS ON OUTPUT FILE
        3  ILLEGAL FILE DEFINITION IN FCB
```

HEALS Logging Program Messages

H0	Abnormal ECFR termination (see ECFR execution report for the cause of the termination).
H1	No hardcore processor error record buffers exist. HEALS Retry statistics are not in the system.
H2	MME GENEWS request to spawn ECFR was denied because HEALS has spawned eight programs; the SNUMB table is full; the memory allocation queue is full; or the requested SNUMB already is in .CRSNB table. (To recover, re-execute HEAL.)
H3	ECF access denied for reason other than NAME NOT IN MASTER CATALOG after a MME GEMORE to request ECF access. Either undetected hardware error or a system software failure occurred. The reason code is in the A-register.
H4	ECF access denied after a MME GEMORE to request ECF access, following initial creation of the ECF by HEALS. Either undetected hardware error or a system software failure occurred. The reason code is in the A-register.
H6	ECF or ESF is released or is a system device. An undetected hardware error occurred, or a system software failure occurred.
H7	ECF physical block size is 40 words. HEAL requires that ECF reside on a device with 64-word blocks.
H8	ESF access denied after a MME GEMORE to request ESF access. Either an undetected hardware error occurred, or a system software failure occurred. The reason code is in the A-register.
H9	ASF access denied after a MME GEMORE to request ASF access. Either an undetected hardware error occurred, or a system software failure occurred.
HB	MPC statistics sampling was requested after the MPC statistics section was released from the HEALS logging program core space. The sample period was zero when HEALS was initially spawned. Respawn HEALS to start a copy of HEALS with MPC logging capability.

HC	Cache memory contents compare error during HEALS cache test after a cache enable by HEALS. This is a hardware error (See Honeywell Error Analysis and Logging Program (HEALS) reference manual for corrective procedure).
HD	HEALS internal channel descriptor block was full. The MPC configuration is too large for the HEALS logging program to handle.
HE	Too little memory space reserved for MPC device statistics. MPC configuration too big for HEALS logging program. If the system is reconfigured, execute the MPC Statistics Display program (MPCD) with a $ SET 22 option. When HEALS is re-executed, the channel descriptor block can be re-created more compactly.
HF	Configuration change caused device statistics space requirements to overflow allocated memory space. To recover, spawn the HEALS Logging program again to initialize HEALS with enough space to handle the new MPC configuration.
HI	Illogical condition detected internally. An undetected hardware error occurred, or a system software failure occurred. This abort may occur if one or more HEALS files were created by an earlier version of the HEALS Logging program. When a new software version is installed, the HEALS catalog and files should be removed by executing a FILSYS activity (see HEALS Catalog and Files Deletion paragraph in the HEALS reference manual) before HEALS is initially executed.
HJ	A cache memory error occurred after cache memory disable was attempted. This may indicate a solid cache memory error and ENA switch ON (processor ENA switch should be OFF).
HK	HEALS request (MME GEFYSE) to create HEALS master catalog was denied. FILSYS abort code (as defined in the FMS manual) is in the A-register.
HL	HEALS request (MME GEFYSE) to create the ECF was denied. FILSYS abort code (as defined in the FMS manual) is in the A-register. For example, the value 401000000000 in the A-register (octal 10 abort code) indicates insufficient file space available on requested device (space on ST1 is requested by HEALS).
HM	HEALS request (MME GEFYSE) to create the ESF was denied. FILSYS abort code (as defined in the FMS manual) is in the A-register.
HW	ECF size not modulo 12 llinks or is less than two links.
HX	Inconsistent ECF size.
HY	HEALS GPR L console verb was entered to enable Exception Processing record transmission to HEALS ECF but Exception Processing collection size is zero.
HZ	HEALS found invalid size field in Exception Processing record.

HEALS Reporter Program Messages

HO	Abnormal HEALS Logging program termination. Examine the HEALS Logging program execution report to determine the cause of the termination.

H1	No hardcore processor error record buffers exist. An undetected hardware error occurred, or a system software failure occurred.
H4	ECF access denied after a MME GEMORE to request ECF access, following initial creation of the ECF by HEALS. Either undetected hardware error or a system software failure occurred.
H6	ECF or ESF is released or is a system device. An undetected hardware error occurred, or a system software failure occurred.
H9	ECFR has been waiting at least three minutes for HEAL to assign ECFR a partial ECF segment. This was caused by the HEAL program being terminated while ECFR was in execution.
HW	ECF size determined by GEFADD is not modulo 12 llinks or is less than two links.
HZ	Last logical Type 03 Exception Processing record from ECF did not fit in physical record.

MPC Statistics Display Program Messages

H1	No main-memory processor error record buffers exist. An undetected hardware error occurred, or a system software failure occurred.
H6	ECF or ESF is released or is a system device. An undetected hardware error occurred, or a system software failure occurred.
H8	ESF access denied after a MME GEMORE to request ESF access. A $ SET 21 or $ SET 22 option was included in MPCD to request ESF access, but ESF did not exist because the HEALS Logging program has not been executed. The denial reason code is in the A-register. If the HEALS Logging program is in execution, this abort indicates an undetected hardware error or a system software failure.
HG	Error during MPC Read/Write memory scan. The MPC configuration table pointed to a channel with a non-MPC type code.
HH	Error during scan to print MPC device statistics. The MPC configuration table pointed to a channel with a non-MPC type code.
HI	Illogical condition detected internally. An undetected hardware error occurred, or a system software failure occurred. This abort may occur if one or more HEALS files were created by an earlier version of the HEALS Logging program. When a new software version is installed, the HEALS catalog and files should be removed by executing a FILSYS activity (see HEALS Catalog and Files Deletion paragraph in the HEALS reference manual) before HEALS is initially executed.

I/O Supervisor Messages

I3	File code not defined. File code specified in I/O sequence is not defined in system; i.e., it is not specified on user's $ FILE or $ TAPE card and it is not a system file.
I5	No file pointer defined in I/O select sequence of MME GEINOS or .GEINOS instruction.
I7	Access beyond file. Seek address in select sequence for sequential file is outside file's boundary.

<u>I-D-S System Messages</u>

ID	I-D-S has aborted program for cause, as indicated by its abort message. (See <u>I-D-S Programmer's Guide</u> reference manual.)

<u>I/O Supervisor Messages</u>

K1	Invalid I/O on device. Program attempted to issue command to a device, which is illegal for that device type.
K2	Bad status return pointer. Status return pointer in select sequence does not point to a location within addressable space of program initiating select sequence.
K3	Invalid file pointer. File pointer to word containing file code addresses location outside program boundaries.
K4	Invalid DCW pointer. DCW pointer addresses location outside program boundaries.
K5	Courtesy call pointer in select sequence addresses location outside program boundaries.
K6	Invalid command for file. Program tried to issue I/O command to file that is invalid for that file type; e.g., linked file command issued to random file.
K7	Two successive TDCW's. IOS does not permit two successive Transfer Data Control Words (TDCW's).

<u>FORTRAN System Message</u>

LK	No $ ENTRY card for this link.

<u>General Loader Program Messages</u>

L1	Missing subroutine required.
L2	NOGO option exercised or Program and Loader overlap.
L3	Fatal error encountered during loading (fatal loading errors are listed in <u>General Loader</u> reference manual).
L4	I/O error abort with same format as File and Record Control GF abort messages.

Dump Analyzer Messages

ME	Memory fault; dump analyzer (.MMDMP) attempted access outside allotted memory space.
NT	Tape with requested density not available.

File and Record Control Message

NT	Attempt to access teletypewriter through File and Record Control without having loaded proper routine, .RTYP option entered as SYMREF on the $ USE card.

Dump Analyzer Messages

OK	User requested program abort of .MMDMP by using $ SET 30 card.
OP	Illegal procedure or command fault in .MMDMP code.
OV	Overflow fault in .MMDMP code.

GMAP Compiler Message

PO	Maximum of 63 levels of MACRO expansion exceeded.

GMAP SALT Aborts

P1	GEMORE denied for memory, A*, or *A
P2	No G* present
P3	Internal error
P4	Illegal EOF from *A

FORTRAN System Messages

Q1	Logical Unit Table overflow.
Q2	Missing Logical Unit Table.
Q3	No space for Logical Unit 6 Buffer.
Q4	Machine error or unexpected error to FORTRAN Compiler.
Q5	FXEM told to take an alternate return but an alternate return name was not supplied.
Q6	Termination of object program execution via FXEM (FORTRAN Execution Error Monitor).

ALGOL System Messages

Q3	Logical Unit 6 not present.
QA	Recursive call to the error processor (i.e., an error has occurred while trying to output an error message).

SORT Program Message

SC	SORT abort; reason indicated on execution report.

COBOL System Message

SM	COBOL Linkage not in stack.

System Library Editor Message

SO	System Editor has encountered an irrecoverable error.

Utility Routine Messages

U1	Control deck error.
U2	Comparison error.
U3	Hardware detected error.

JOVIAL System Messages

V0	Compiler error abort. Fatal error was encountered when processing direct code.
V1	Compiler error abort. An alternate construction of the indirect statement will probably allow it to compile.
V2	Object program abort. The JOVIAL Compiler generated a MME GEBORT when replacing an erroneous JOVIAL statement.
V3	Compiler memory exhausted abort. Compiler needs more storage and its requests for additional memory have been denied. Therefore, additional space was not available to complete compilation.
V4	Compile activity has completed and preparation for loader activity (binding the object H*) has aborted because not enough memory is available to satisfy memory specification on RUN command. H* file will not be bound but, if on the RUN command a request was made to save a C* file, the C* file for the object JOVIAL program will be created. The object program will not be placed in execution.
V5	Compiler error: Fatal error; an illegal construct has been encountered.

Operator-Initiated Abort Messages

X1	Operator deleted job from control stack.
X2	Operator aborted job in execution.

FORTRAN Messages

NOTE: The abort code Y1 is always displayed as the reason code for any abort. The panel reveals the specific reason code (see codes in parentheses of following descriptions) in the upper 18 bits of the Q-register.

Y1 (X1)	Compiler space management module has unsuccessfully attempted to allocate contiguous memory block for internal table. Rerun with DUMP option and $ SYSOUT card for file code *F. Return dump to Honeywell Field Support - Phoenix.
Y1 (X2)	Compiler has attempted to execute request for additional core space more than 10 consecutive times (initial memory space plus maximum of 30K). Increase allocation via $ LIMITS card or via "CORE=" option on TS RUN.
Y1 (X3)	GCOS has denied compiler request for additional memory space for internal tables. Increase allocation via $ LIMITS card or via "CORE=" option on TS RUN.
Y1 (O3)	Expression being handled has tree structure depth greater than 64. Expression must be divided.
Y1 (O4)	Rerun with DUMP option and $ SYSOUT card for file code *F. Return dump to Honeywell Field Support - Phoenix.
Y1 (P4)	Unrecoverable error occurred in code generator; error message will print following source statement causing abort. Rerun with DUMP option and $ SYSOUT card for file code *F. Return dump to Honeywell Field Support - Phoenix.

General Remote Terminal Supervisor (GRTS) Messages

41	Invalid I/O status pointer in MME GEROUT.
43	Invalid DCW pointer or illegal character count.
72	Invalid MME GEROUT operation code.
73	PAT area exhausted.
74	Output record too large for terminal type.
75	Undefined remote terminal station identification or duplicate remote inquiry name.
76	Invalid MME GEROUT program identification or program number illegal.

SYSTEM ABORT MESSAGE TEXTS

Octal Code	Text
00	NO REASON SPECIFIED
01	CANNOT LOAD SSA MODULE
02	SSA MODULE CHECKSUM ERROR
03	CANNOT FIND SSA MODULE
05	I8-RUN TIME EXHAUSTED
06	I2-MME GEROAD IN CC (Courtesy Call)
07	I6-GEENDC BUT NOT IN CC
10	PRIMARY MAILBOX ERROR
11	CONNECT OR T.I. ERROR
12	IOC DATA-SERVICE ERROR
13	IOC MEMORY ADDRESS FAULT
14	IOC NOT MASTER MODE FLT.
15	MEMORY PARITY DURING I/O
16	IOC NOT CTL. PROC. FAULT
17	IOC MEM. PROTECT FAULT
20	MEMORY TIME-OUT IN I/O
21	OTHER IOC-MEMORY ERROR
22	F0-MEMORY ADDRESS FAULT
23	I1-IMPROPER MME ADDRESS
24	F1-FAULT TAG FAULT
25	F2-COMMAND FAULT
26	F3-DERAIL FAULT
27	F4-LOCKUP FAULT
30	F6-MEMORY PARITY FAULT
31	F7-UNDEFINED OP. FAULT
32	F9-OVERFLOW/UNDERFLOW
33	I0-DIVIDE CHECK FAULT
34	F8-OP NOT COMPLETE FAULT
35	I/O OUT OF MEMORY BOUNDS
36	I3-FILE CODE NOT DEFINED
37	I5-NO GEINOS FILEPOINTER
40	I7-ACCESS BEYOND FILE
41	K2-BAD STATUS RET. PTR.
42	K3-INVALID FILE POINTER
43	K4-INVALID DCW POINTER
44	K5-BAD COURTESY CALL PTR
45	K6-BAD I/O COMMAND-FILE
46	K7-TWO SUCCESSIVE TDCWS
47	K1-INVALID I/O ON DEVICE
50	OPER REQ ABORT NODUMP
51	OPER REQ ABORT
52	M4-N4-I/O LIM. CALL/SAVE/RSTR
53	M6/N7-I/O ERR. CALL/SAVE/RSTR
54	M5-NO PAT FOR CALL/SAVE/RSTR
55	M1/N5-BAD DEV. CALL/SAVE/RSTR

Octal Code	Text
56	NON-RANDOM GECALL/GESAVE/GERSTR FILE
57	N8-GESAVE FILE IS FULL
60	M6-CALL/SAVE/RSTR CHECKSUM
61	LOW GESAVE/GERSTR ORIGIN USED
62	N4-ZERO GESAVE WORD CT.
63	M2/M3-CALL/RSTR NAME MISSING
64	N8-CALL OUT OF FILE SPAN
65	IMPROPER GECOS CALL
66	GEMORE - ZERO FILECODE
67	MO-BAD GEMORE PARAMETER
70	NO ROOM IN PAT - GEMORE
71	M7-NOT DISC/DRUM GEMORE
72	R1-BAD GEROUT OP. CODE
73	R4-GEROUT PAT EXHAUSTED
74	RMT TERMINAL RECORD SIZE
75	UNDEF REMOTE STATION ID
76	INVALID GEROUT PROG. ID
77	O -OUTPUT LIMIT EXCEEDED
100	SYSOUT RECORD SIZE ERROR
101	SYSOUT SEEK ERROR
102	SYSOUT ALLOCATION ERROR
103	O1-BAD SYOT STATUS PTR.
104	O1-BAD SYOT BUFFER PTR.
105	O1-SYOT BUFFER LIMITS
106	EP-IRRECOVERABLE I/O ERR
107	K4-INVALID SEEK DCW I/O
110	BAD GESYOT MEDIA CODE
111	N3-ROLLBACK NOT POSSIBLE
112	TSS REQUESTED TERM
113	GEMORE DUPLICATE FILE
114	INVALID I/O SCT POINTER
115	IMPERMISSIBLE PERM WRITE
116	IMPERMISSIBLE PERM READ
117	D2-IDS RECORD TOO LONG
120	SYSOUT STORAGE EXHAUSTED
121	INVALID MME PARAMETER
122	LOST EXTRA SSA CONTENTS
123	J3-ALOC DELETED JOB
124	BAD BUFFER ADDR-GENEWS
125	GUESSING ON PRMFL GEMORE
126	NO USERID-PRMFL GEMORE
127	OPER REQ ABORT DUMP
130	CANNOT MOVE JOB IN CORE
131	IOM CHAN UNEXPECTED PCW
132	IOM CHAN IMPROPER INSTR.

Octal Code	Text
133	IOM CHAN IMPROPER DCW
134	IOM CENT IPW TALLY RNOUT
135	IOM CENT TWO TDCWS
136	IOM CENT BOUNDARY ERROR
140	IOM CENT IDCW RES MODE
141	IOM CENT CHAR POS/SZ DIS
142	BACKDOOR F.C. UNKNOWN
143	INVALID FILE TO BACKDOOR.
144	IDS ADDRESS OUT OF BOUND
145	SECUR-SCC INVALID
146	FILE ALLOCATION ERROR
147	SECUR-SCC INVAL FOR USER
150	SECUR-SCC INVAL FOR LINE
151	SECUR-DUPLICATE REPORT
152	SECUR-INVALID DCW STRING
153	SECUR-INVALID FUNCT CODE
154	NO LINKS FOR SWAP FILE
155	DATA ACCESS SYSTEM ABORT
156	DATA MANAGEMENT SYSTEM PROTECTION FAILURE
157	BSS (Bulk Store Subsystem) MAIN MEMORY BOUNDARY ERROR
160	BSS NO SYSTEM RESPONSE
161	BSS CONNECT HALT
162	ISP SYSTEM TERMINATION
163	ILLEGAL EIS DATA
164	I/O CHANNEL DEACTIVATED
165	USER ABORT REQUEST
166	F5-CACHE PARITY FAULT
167	INVALID MME GERELS REQST
170	INVALID SEARCH RULE FILE
171	BAD SEARCH RULE RECORD
172	INVALID PROCEDURE FILE
173	BAD PROCEDURE FILE CODE

APPENDIX B

CHARACTER SET TRANSLITERATION TABLES

The system can accept source and data cards which have been punched in the IBM or the G-225 character sets. The transliteration is accomplished by placing the proper $ INCODE control ahead of the cards punched in the IBM or G-225 character set as follows:

1	8	16	
$	INCODE	IBMF	(For IBM FORTRAN)
$	INCODE	IBMC	(For IBM COBOL)
$	INCODE	IBMEL	(For IBM 360 extended language)
$	INCODE	GE225	(For G-225)
$	INCODE	IBMPL	(For IBM PL/I)

The transliteration tables for the above character sets are included. Representation of characters in memory storage is after transliteration.

TRANSLITERATION TABLE FOR IBM FORTRAN (IBMF) CHARACTER SET

IBMF Character	IBMF Card Punch	Main Memory	Series 60/Series 6000 Character
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	10	8
9	9	11	9
none	8-2	12	[
=	8-3	75	=
'(apostrophe)	8-4	57	'(apostrophe)
none	8-5	13	#
none	8-6	16	>
none	8-7	17	?
blank	blank	20	blank
A	12-1	21	A
B	12-2	22	B
C	12-3	23	C
D	12-4	24	D
E	12-5	25	E
F	12-6	26	F
G	12-7	27	G
H	12-8	30	H
I	12-9	31	I
+	12	60	+
.	12-8-3	33	.
)	12-8-4	55	)
none	12-8-5	35	(
none	12-8-6	36	<
none	12-8-7	37	\
none	11-0	40	↑
J	11-1	41	J
K	11-2	42	K
L	11-3	43	L
M	11-4	44	M
N	11-5	45	N
Ø	11-6	46	O
P	11-7	47	P
Q	11-8	50	Q
R	11-9	51	R
-	11	52	-
$	11-8-3	53	$
*	11-8-4	54	*

IBMF Character	IBMF Card Punch	Main Memory	Series 60/Series 6000 Character
none	11-8-5	55	)
none	11-8-6	56	;
none	11-8-7	57	'(apostrophe)
none	12-0	60	+
/	0-1	61	/
S	0-2	62	S
T	0-3	63	T
U	0-4	64	U
V	0-5	65	V
W	0-6	66	W
X	0-7	67	X
Y	0-8	70	Y
Z	0-9	71	Z
none	0-8-2	72	←
,(comma)	0-8-3	73	,(comma)
(	0-8-4	35	(
none	0-8-5	75	=
none	0-8-6	76	"
none	0-8-7	77	!

TRANSLITERATION TABLE FOR IBM COBOL (IBMC) CHARACTER SET

IBMC Character	IBMC Card Punch	Main Memory	Series 60/Series 6000 Character
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	10	8
9	9	11	9
none	8-2	12	[
=	8-3	75	=
'	8-4	76	"
none	8-5	13	#
none	8-6	16	>
none	8-7	17	?
blank	blank	20	blank
A	12-1	21	A
B	12-2	22	B
C	12-3	23	C
D	12-4	24	D
E	12-5	25	E
F	12-6	26	F
G	12-7	27	G
H	12-8	30	H
I	12-9	31	I
+	12	60	+
.	12-8-3	33	.
)	12-8-4	55	)
none	12-8-5	35	(
none	12-8-6	36	<
none	12-8-7	37	\
none	11-0	40	↑
J	11-1	41	J
K	11-2	42	K
L	11-3	43	L
M	11-4	44	M
N	11-5	45	N
O	11-6	46	O
P	11-7	47	P
Q	11-8	50	Q
R	11-9	51	R
-	11	52	-
$	11-8-3	53	$
*	11-8-4	54	*

IBMC Character	IBMC Card Punch	Main Memory	Series 60/Series 6000 Character
none	11-8-5	55	)
none	11-8-6	56	;
none	11-8-7	57	'(apostrophe)
none	12-0	60	+
/	0-1	61	/
S	0-2	62	S
T	0-3	63	T
U	0-4	64	U
V	0-5	65	V
W	0-6	66	W
X	0-7	67	X
Y	0-8	70	Y
Z	0-9	71	Z
none	0-8-2	72	←
,(comma)	0-8-3	73	,(comma)
(	0-8-4	35	(
none	0-8-5	75	=
none	0-8-6	76	"
none	0-8-7	77	!

TRANSLITERATION TABLE FOR G-225 CHARACTER SET

G-225 Character	G-225 Card Punch	Main Memory	Series 60/Series 6000 Character
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	10	8
9	9	11	9
none	8-2	12	[
#	8-3	13	#
@	8-4	14	@
underline___	8-5	52	-
=	8-6	75	=
none	8-7	17	?
blank	blank	20	blank
A	12-1	21	A
B	12-2	22	B
C	12-3	23	C
D	12-4	24	D
E	12-5	25	E
F	12-6	26	F
G	12-7	27	G
H	12-8	30	H
I	12-9	31	I
+	12	60	+
.	12-8-3	33	.
]	12-8-4	34	]
none	12-8-5	35	(
none	12-8-6	36	<
none	12-8-7	37	\
-0	11-0	40	↑
J	11-1	41	J
K	11-2	42	K
L	11-3	43	L
M	11-4	44	M
N	11-5	45	N
O	11-6	46	O
P	11-7	47	P
Q	11-8	50	Q
R	11-9	51	R
-	11	52	-
$	11-8-3	53	$
*	11-8-4	54	*

G-225 Character	G-225 Card Punch	Main Memory	Series 60/Series 6000 Character
none	11-8-5	55	)
none	11-8-6	56	;
none	11-8-7	57	'(apostrophe)
+0	12-0	60	+
/	0-1	61	/
S	0-2	62	S
T	0-3	63	T
U	0-4	64	U
V	0-5	65	V
W	0-6	66	W
X	0-7	67	X
Y	0-8	70	Y
Z	0-9	71	Z
none	0-82	72	←
,(comma)	0-8-3	73	,(comma)
%	0-8-4	74	%
(	0-8-5	12	]
)	0-8-6	34	[
none	0-8-7	77	!

TRANSLITERATION TABLE FOR IBM 360 EXTENDED LANGUAGE (IBMEL) CHARACTER SET

IBMEL Character	IBMEL Card Punch	Main Memory	Series 60/Series 6000 Character
0	0	00	0
1	1	01	1
2	2	02	2
3	3	03	3
4	4	04	4
5	5	05	5
6	6	06	6
7	7	07	7
8	8	10	8
9	9	11	9
:	2-8	15	:
#	3-8	13	#
@	4-8	14	@
'(apostrophe)	5-8	57	'(apostrophe)
=	6-8	75	=
"	7-8	76	"
blank	blank	20	blank
A	12-1	21	A
B	12-2	22	B
C	12-3	23	C
D	12-4	24	D
E	12-5	25	E
F	12-6	26	F
G	12-7	27	G
H	12-8	30	H
I	12-9	31	I
&	12	32	&
¢	12-8-2	37	\
.	12-8-3	33	.
<	12-8-4	36	<
(	12-8-5	35	(
+	12-8-6	60	+
logical OR|	12-8-7	37	\
none	11-0	40	↑
J	11-1	41	J
K	11-2	42	K
L	11-3	43	L
M	11-4	44	M
N	11-5	45	N
O	11-6	46	O
P	11-7	47	P
Q	11-8	50	Q
R	11-9	51	R
-	11	52	-
!	11-8-2	37	\
$	11-8-3	53	$
*	11-8-4	54	*

IBMEL Character	IBMEL Card Punch	Main Memory	Series 60/Series 6000 Character
)	11-8-5	55	)
;	11-8-6	56	;
logical NOT ¬	11-8-7	20	blank
none	12-0	60	+
/	0-1	61	/
S	0-2	62	S
T	0-3	63	T
U	0-4	64	U
V	0-5	65	V
W	0-6	66	W
X	0-7	67	X
Y	0-8	70	Y
Z	0-9	71	Z
none (0-8-2)	0-2-8	20	blank
,(comma)	0-3-8	73	,(comma)
%	0-4-8	74	%
underscore__	0-5-8	20	blank
>	0-6-8	16	>
?	0-7-8	17	?

NOTE: The IBMPL transliteration allowed on the $ INCODE and $ DATA cards differs from the above transliteration set in the following characters:

Character	IBMEL	IBMPL	Card Punch
logical OR	\ (37)	! (77)	12-8-7
logical NOT	blank (20)	↑ (40)	11-8-7
underscore	blank (20)	← (72)	0-5-8

INDEX

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	SERIES 60(LEVEL 66),6000 GENERAL COMPREHENSIVE OPERATING SUPERVISOR (GCOS)	ORDER NO.	DD19-01
		DATED	AUGUST 1980

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME ______________________ DATE ____________

TITLE ______________________

COMPANY ______________________

ADDRESS ______________________

____

-EASE FOLD AND TAPE—
OTE: U. S. Postal Service will not deliver stapled forms

CUT ALONG LI

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell