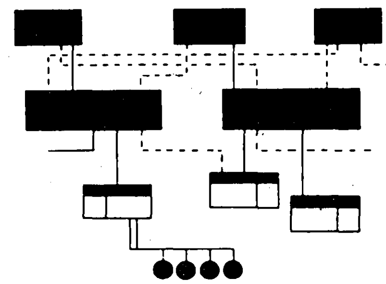
GE-625/635 GECOS-III Startup

Information Systems

Information Systems Equipment

SOFTWARE MAINTENANCE DOCUMENT



GENERAL ELECTRIC

CPB-1489

CPB-1489

GE-625/635 GECOS-III Startup

SOFTWARE MAINTENANCE DOCUMENT

March 1968

INFORMATION SYSTEMS

GENERAL ELECTRIC

PREFACE

This manual describes the implementation of Startup processing for the GE-625/635 General Comprehensive Operating Supervisor (GECOS-III).

Additional software maintenance documents are as follows:

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

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

GE-625/635 GECOS-III Dispatcher and Peripheral Allocation, CPB-1491

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

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

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

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

GE-625/635 GECOS-III Termination and System Output, CPB-1469

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

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

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

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

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

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

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

Suggestions and criticisms relative to form, content, purpose, or use of this manual are invited. Comments may be sent on the Document Review Sheet in the back of this manual or may be addressed directly to Documentation Standards and Publications, B-90, Processor Equipment Department, General Electric Company, 13430 North Black Canyon Highway, Phoenix, Arizona 85029.

CONTENTS

ILLUSTRATIONS

1. SYSTEM INITIALIZATION

System initialization consists essentially of two procedures. The first is the physical configuration of the system modules during which time various switches are set on the IOC, processor, and memory controller T&O panels. Once the hardware is configured, the Startup program is used to initialize the resident software files, to load GECOS permanent modules into memory, and to describe to GECOS the system environment in which it is to operate.

The GECOS system tapes, distributed by CED, are in system loadable format, described in the GE-625/635 System Editor reference manual, CPB-1138. As shown in Figure 1, permanent modules are placed on secondary system storage (ST2) and temporary modules are stored on primary system storage (ST1). Secondary system storage is defined as the slower device if a choice of devices is available, whereas primary storage is defined as the faster device. For example, if both disc and drum are available, the faster device (drum, in this case) should be used as primary system storage for GECOS temporary modules and any other high-use system software.

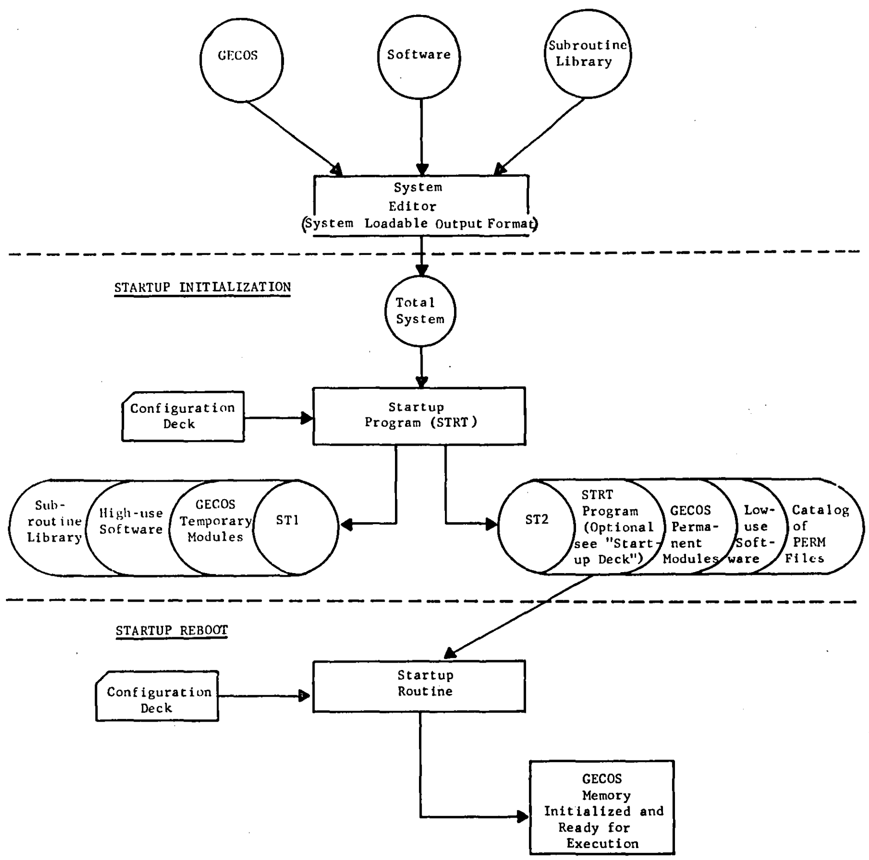


Figure 1. System Startup

INITIALIZATION SETUP PROCEDURE

The specific setup procedure used to initialize GECOS is dependent upon whether GECOS is operating in a uniprocessing or multiprocessing environment and, further, is dependent upon the specific hardware configuration. For this reason, brief descriptions of the functions of each switch on the processor, memory controller, and IOC are given prior to descriptions of typical uniprocessing and multiprocessing setup procedures.

IOC T & O Panel Switch Functions

Figure 2 is a schematic of the IOC T&O panels showing the various switches. Also shown are the T&O panel switches for the memory controller and the IOC, the functions of which are described in the following sections.

IOC Switches

The functions of the IOC switches listed below are described in this section:

PRIMARY MAILBOX
CONNECT SELECT
PORT SELECT
RECEIVE REGISTER SELECT
TRANSFER FROM ADDRESS (IOC-B) LOAD ADDRESS (IOC-C)
TRANSFER FROM DATA (IOC-B) LOAD DATA (IOC-C)

The PRIMARY MAILBOX switches allow relocating the primary mailbox for any given IOC. The switches are used to place the IOC control words (IOC Mailboxes containing 256 words) at various addresses (0 Mod 256) in lower memory. Within the 256-word block are secondary mailboxes controlled by software. The four switches (L29, L28, L27, L26) represent the four high-order bits of a 12-bit address.

The CONNECT SELECT switch (rotary, 4-position) is used to select the memory in which the primary mailbox resides (that is, the memory from which the IOC will accept a connect and the memory in which the IOC will set interrupts).

The PORT SELECT switches provide a means of determining to which memory a given address must be routed. The four sets of switches (one set for each IOC port: A, B, C, and D) are interpreted exactly like the CHANNEL ADDRESS REASSIGN on the processor (see following section entitled "Processor Switches").

(On an IOC-C, both RECEIVE REGISTER switches are located on the right side of the T&O panel. The locations given below are those on an IOC-B.)

The two RECEIVE REGISTER SELECT switches are used to select a register to be loaded from the panel. The leftmost switch will transfer the data contained in 16 thumbwheels to the right of the switch. In the U position, the switch selects the data as though the data were coming from memory. The rightmost switch is used to select either the Buffer Address Register (BAR) or the Program Counter (PCT).

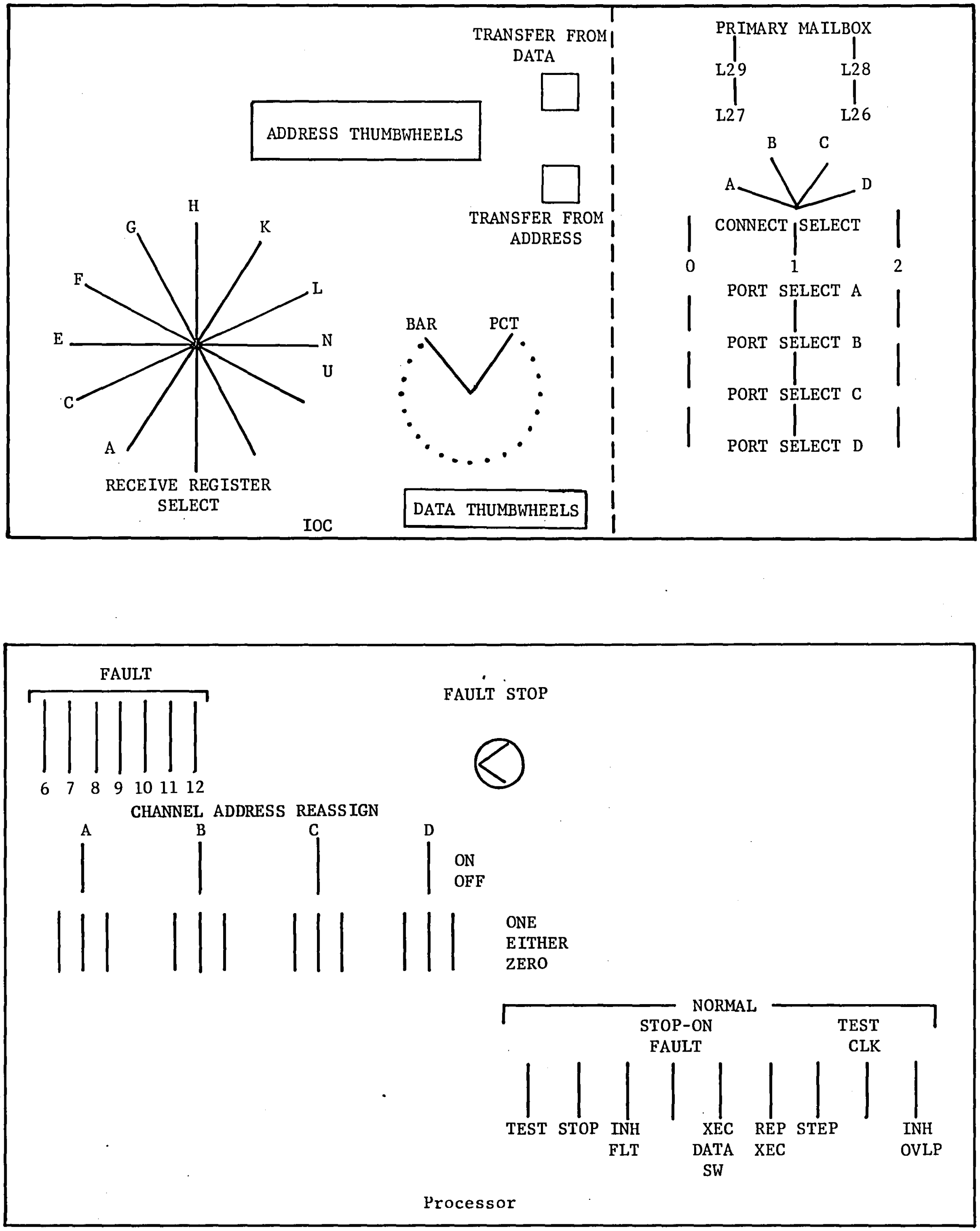


Figure 2. Memory Controller, Processor, IOC Panels

CPB-1489

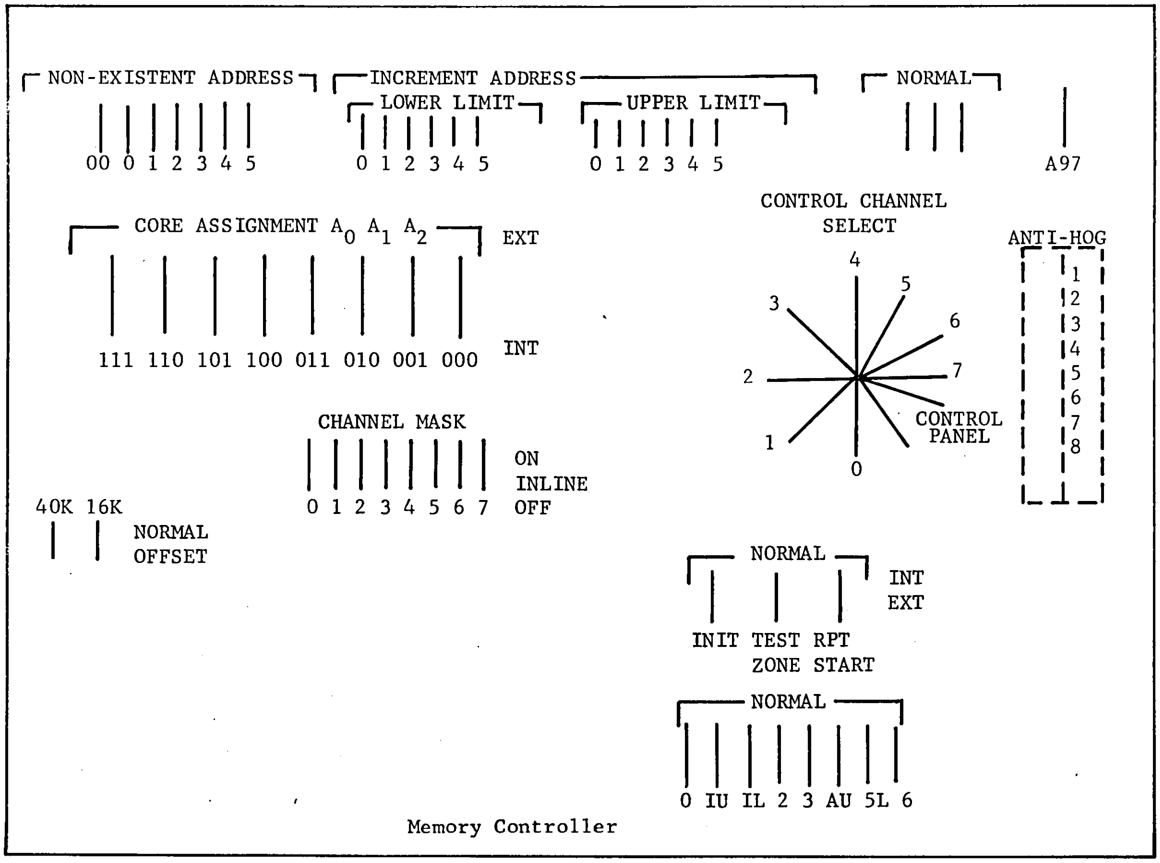


Figure 2. (continued)

CPB-1489

The TRANSFER FROM ADDRESS switch transfers the settings of the ADDRESS thumbwheel switches to the register selected by the RECEIVE REGISTER SELECT rotary switch, which is located at the left of the ADDRESS switches.

The TRANSFER FROM DATA switch transfers the settings of the DATA THUMBWHEEL switches to the register selected by the RECEIVE REGISTER SELECT rotary switch, which is located at the left of the DATA switches.

Processor Switches

The functions of the processor switches listed below are described in this section:

FAULT
CHANNEL ADDRESS REASSIGN
TEST
STOP
INHIBIT FAULT
STOP-ON FAULT
FAULT STOP SELECTOR SWITCH
EXECUTE DATA
EXECUTE Pushbutton
REPEAT EXECUTE
STEP
ADV MAJOR CYCLE
TEST CLOCK
INHIBIT OUTPUT

The set of FAULT switches is used in servicing machine-detected faults. Through these, the machine location in which the 16 two-word fault cells (Interrupt Vectors) begin is determined. The seven switches represent the seven high-order bits of a 12-bit (4k) address. The remaining bits are assumed zero. A switch in the UP position (1 state) turns its bit on, and a DOWN switch (0 state) turns its bit off.

The CHANNEL ADDRESS REASSIGN switches are used to determine to which memory controller a given address is to be routed (see also discussion of memory controller switches). There are four sets of switches; one set for each possible port where a memory controller may reside. For each port, there is a single switch (A, B, C, D) to turn that port ON or OFF. When OFF, the indication is that no cabling or communication exists for that port. Actual address routing is provided by the three lower switches. These switches have three positions: ON, EITHER, and ZERO and represent the three most significant bits of a full 18-bit address. All other bits are ignored. When address determination is required, the first three bits are checked against the switch settings. The address will be routed to the memory controller connected to the port whose switches correspond to the value of the first three bits.

The TEST switch activates all other switches within the same group (STOP, INHIBIT FAULT, STOP ON FAULT, etc.). When in the NORMAL (down) position, the switches are activated.

The STOP switch is used to stop the processor by forcing a DIS (Delay until Interrupt Signal). It should be noted that any outstanding interrupts will cause the processor to resume operation once the switch is returned to its normal state.

The INHIBIT FAULT switch allows faults to be ignored. When this switch is NORMAL (down), none of the following faults will be processed:

Out-of-Bounds fault (MEM)
Lockup fault
Operation-Not-Complete fault
Startup fault
Timer Runout fault
Connect fault
Shutdown fault

When the INHIBIT FAULT switch is in the NORMAL position (up), all of the above faults are processed.

The STOP-ON FAULT switch is used to stop the processor upon the detection of the fault indicated by the FAULT STOP switch (see below). The processor will be stopped before the pair of instructions in that fault vector are executed.

The FAULT STOP selector switch is used in conjunction with the STOP-ON FAULT switch (must be DOWN) to select a specific fault indicated by the rotary switch. When used in this manner, any of the following faults may be selected:

ONC - Operation Not-Complete
PAR - Parity
ZOP - Zero Operation Code
LUF - Lockup Fault
CMD - Command
TAG - Fault TAG
MEM - Memory
TRO - Timer Runout

In addition to those faults listed above, the STOP-ON FAULT selector switch may be moved to the ALL position. In this position all faults (including master mode entries (MME) and derails (DRL)) will cause the processor to stop before the fault is executed.

The EXECUTE DATA SWITCHES switch causes execution of whatever is in the 36 DATA switches when the EXECUTE pushbutton is depressed and the EXECUTE DATA SWITCHES switch is down. The data will be transferred to both the even and odd instruction and address registers. (Instead of executing the two instructions at the Execute Fault address the instruction contained in the DATA switches will be executed by the processor.)

The EXECUTE pushbutton will activate the execution of an Execute Fault if the TEST switch is in the NORMAL (down) position and the EXECUTE DATA SWITCHES switch is in the NORMAL position. If the EXECUTE DATA SWITCHES switch is in the NORMAL position, the processor will execute the instruction set up on the DATA switches.

The REPEAT EXECUTE switch enables the processor to be tied into a program loop for maintenance purposes.

The STEP switch allows the processor to be single stepped through a normal program flow, one instruction at a time. Single stepping is done by depressing the ADV MAJOR CYCLE pushbutton.

The ADV MAJOR CYCLE pushbutton allows manual stepping from one instruction to the next. It should be noted that instructions involving indirect and tally operations cannot be stepped if the memory controller TEST switch is in the NORMAL (up) position.

The TEST CLOCK is used for maintenance purposes to add (or subtract) margin times to (or from) critical delay lines in the processor so that aging components may be identified.

The INH OVLP switch prevents overlapping instructions. Normally, the processor gets two instructions and, while one is being executed, the next is being fetched or the processor is looking up the next two. The switch inhibits this overlap.

Memory Controller Switches

The functions of the memory controller switches listed below are described in the following paragraphs:

NON-EXISTENT ADDRESS
CONTROL CHANNEL SELECT
CORE ASSIGNMENT
40K and 16K OFFSET/NORMAL switches
CHANNEL MASK
TEST, INITIALIZE, RPT START
ZONE
Combined switch usage for clearing memory to zeroes
Memory ANTI-HOG

The set of NON-EXISTENT ADDRESS switches determines the upper memory boundary within this controller. Any address received beyond the setting will cause a memory or address fault. Starting at Switch 2 and working left, the values are: Switch 2 = 32k, Switch 1 = 64k, Switch 0 = 128k, and Switch 00 = 256k. For the configuration defined in Figure 3, the NON-EXISTENT ADDRESS switches would be set as follows:

Memory Controller	NON-EXISTENT ADDRESS switch							Maximum Address Contained in Memory Controller
	00	0	1	2	3	4	5	
0	0	0	0	1	0	0	0	32k
1	0	0	1	0	0	0	0	64k

If a third memory controller were present, the NEA switches on it would be set at NEA 0 and 1 equal to 1; the rest set at zero. Memory controller 0 would be set at NEA 1 and 2 equal to 1 and zero, respectively, and memory controller 1 switches 0 and 1 set at 1 and zero, respectively.

The CONTROL CHANNEL SELECT switch determines which processor will receive interrupts and, therefore, which processor is the control processor. The switch indicates the memory port to which a processor is connected. Although more than one processor may be connected to a memory controller, only one processor (the control processor) may respond to an interrupt.

The CORE ASSIGNMENT switches consist of eight 2-position (EXTERNAL and INTERNAL) assignment switches which interpret address bits A_0, A_1, and A_2, as specified by the switch position settings, and connect the correct core to the memory controller for a given address. Each switch represents one of the eight decoded states of A_0, A_1, A_2. Figure 3 is a block diagram of a two memory controller, 128k setup.

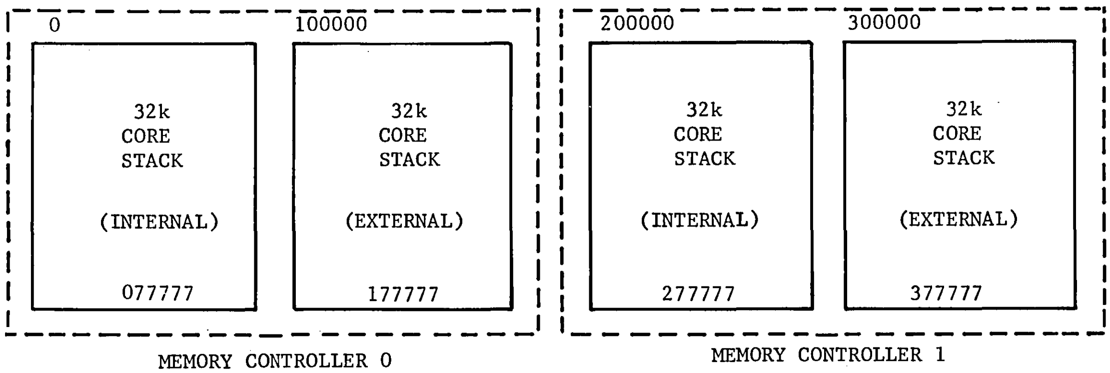


Figure 3. Two Memory Controllers, 128k Setup

Within any one controller there is normally a memory capacity of 32k to 128k. Up to 64k of this memory is physically located within the controller cabinet and is referred to as "internal" memory. An additional capability of up to 64k is contained in an adjacent cabinet and is referred to as "external" memory.

The CORE ASSIGNMENT switches determine which addresses are in internal memory and which are in external memory. Each of the eight switches decodes the three most significant bits of an address and select where that address may be. Normally, all CORE ASSIGNMENT switches for addresses outside the range of a given controller are placed in the external position except for the following restraint required by the hardware: the CORE ASSIGNMENT switch for address field 000 must, in any controller, point to an existing memory in that controller. This restriction means that if no external memory exists for a given controller, the core assignment switch 000 must be placed in the internal position.

Again, with reference to Figure 3, the CORE ASSIGNMENT switches must be set as follows to properly address memory:

	CORE ASSIGNMENT switch							
Memory Controller	111	110	101	011	010	001	000	
0	E	E	E	E	E	E	I	
1	E	E	E	E	I	E	E	(See note)

E = External; I = Internal

NOTE: Switch 000 for memory controller 1 may be placed in the EXTERNAL position because that controller does include an external memory.

The processor or IOC, as previously set up, routes the first 64k addresses (000000-177777) to memory controller 0 which, in turn, routes the first 32k (000000-077777) to its internal memory and the second 32k (100000-177777) to its external memory. The second 64k (200000-377777) which are transmitted to memory controller 1 are distributed with the third 32k (200000-277777) to internal memory and the last 32k (300000-377777) to external memory.

The same procedure is applied throughout the entire range of addresses available in any configuration of the GE-625/635.

The 40k and 16k OFFSET/NORMAL switches are used by the memory controller to provide occasionally required functions for memory address reassignment within one memory unit.

The 40k name derives from requirements for that switch with 40k memory units used on early GE-625 systems. The switch actually provides relocation by a 32k factor. The only requirement for changing the 40k switch from NORMAL position to offset position occurs when the 40k memory unit is utilized on a system. The 40k unit, by definition, can be fully utilized only when it contains the highest system addresses. The 40k switch, in offset position, is used only when the 40k block of addresses starts at an odd multiple of 32k.

The 16k switch is left in the NORMAL position. The only deviation from this practice occurs during execution of diagnostic programs for the memory and is changed under direction of that diagnostic. Under the GECOS environment, this switch should remain in the NORMAL position.

Under conditions where the system is required to maintain operation with a portion of its memory malfunctioning, it is generally possible to reconfigure to exclude only the inoperable area from the system. A table of realizable configurations is too voluminous to be shown here, but the following set of rules can be applied.

1. Determine the exact address area which is inoperable. This address will be continually moved to higher system addresses in the following steps, so it is necessary to keep track of its location.

2. Assign memory controllers (using processor and IOC configuration switches) to place the inoperable area as a part of the memory controller with the highest system absolute addresses.

3. In the memory controller with an inoperable area, reassign with the CORE ASSIGNMENT (Internal/External) switches the memory unit with that inoperable area to the highest address range. This places the inoperable area in the last 32k or 64k of system addresses.

4. Determine the area of inoperability and size of memory unit in which it is contained and proceed to indicated steps in order shown below:

   (a) 32k memory, 0k-16k inoperable; proceed to Steps 6 and 7
   (b) 32k memory, 16k-32k inoperable; proceed to Step 7
   (c) 32k memory, 0k-32k inoperable; proceed to Step 7
   (d) 64k memory, 0k-16k inoperable; proceed to Steps 5, 6, and 7
   (e) 64k memory, 16k-32k inoperable; proceed to Steps 5 and 7
   (f) 64k memory, 32k-48k inoperable; proceed to Steps 6 and 7
   (g) 64k memory, 48k-64k inoperable; proceed to Step 7
   (h) 64k memory, 0k-32k inoperable; proceed to Steps 5 and 7
   (i) 64k memory, 32k-64k inoperable; proceed to Step 7
   (j) 64k memory, 0k-64k inoperable; proceed to Step 7

5. Previous steps have placed the inoperable area within the next to last of system addresses. Place the 40k switch in the offset position to move the inoperable area into the last 32k.

6. Previous steps have placed the inoperable area within the next to last 16k of system address. Place the 16k switch in the OFFSET position to move the inoperable area into the last 16k.

7. Next, determine, in 4k increments counting from the highest address, the exact extent of inoperable area. This will provide the necessary information to set the NEA switches to provide the maximum amount of operable memory operating on-line.

The CHANNEL MASK switches are eight 3-position (ON, IN-LINE, OFF) switches, one per memory channel, used to control the recognition of a channel interrupt present signal. When a switch is ON, the module connected to that port is masked out of the system and no communication can be made to it. When a switch is OFF, communication is always permissible to that module. The IN-LINE position indicates that the mask is alterable by software and is under control of the channel interrupt mask register. It may be turned OFF or ON as the SMCM command sets the register to 0 or 1 respectively.

TEST, INITIALIZE, and RPT-START-EXTERNAL switches are used as test switches. The NORMAL-TEST switch, in the TEST position, enables maintenance panel control. In the NORMAL position it provides a master lockout of the following switches.

INTERRUPT pushbutton	DP/REWRITE
NORMAL-RPT START-EXTERNAL	STOP ON ADDRESS
NORMAL-INITIALIZE	STOP ON FAULT

The NORMAL-INITIALIZE switch, in the INITIALIZE position, stops all memory action and clears the Execute Interrupt register and Access Request register (NORMAL-TEST switch must be in TEST position). The NORMAL-RPT START switch, in the Internal and External RPT START positions, provides the option of submitting an internal or external frequency for manual operation of the INTERRUPT pushbutton (NORMAL-TEST switch must be in the NORMAL position).

The eight 2-position (binary 1 and 0) ZONE select switches are used to designate and/or select the characters (6- or 9-bit) to be written with the CWR command and rewrite portion of the RAR command during off-line (TEST) operations.

As a rapid means of clearing memory to all zeroes, the following steps may be used:

1. TEST/NORMAL to TEST
2. INITIALIZE and return to NORMAL (Suggested, not mandatory)
3. COMMAND to 20_8 (Clear/Write-Single Precision)
4. ZONES to ONES*
5. DATA to ZEROES
6. INCREMENT ADDR/NORMAL to INCREMENT*
7. A17 CONTROL to COUNT*
8. INCREMENT LOWER LIMIT to lowest address in controller (4k increments)*
9. INCREMENT UPPER LIMIT to highest address in controller (4k increments)*
10. RESET INCREMENT ADDRESS (push)
11. NORMAL/REPEAT START to INTERNAL for about 1 second, then NORMAL
12. TEST/NORMAL to NORMAL

In a multiprocessor system, some method must be used to assure equal access of memory by like devices (such as processors) when these devices are connected to the same memory controller. The method used to guarantee equal access is the implementation of the ANTI-HOG SWITCHES. These switches, located on a PWB inside the memory controller door on the lower righthand side, are similar to the eight channel mask switches (that is, ports 0 through 7). A switch in the UP position indicates that the anti-hog circuitry is off for those ports. A switch in the MIDDLE or DOWN position indicates that the anit-hog circuitry is on. (The DOWN position is the recommended position since this position is easier to identify than a switch in the MIDDLE position.)

The ANTI-HOG SWITCHES allow memory access to a group of consecutive ports on the basis of a built-in priority system. The priority system is such that groups of devices on a higher-numbered memory port have a lower priority than those device groups located on lower-numbered ports. Between anti-hog groups, access requests to the higher priority group (see Figure 4) must be completely satisfied before any requests to the lower priority group are acknowledged. In addition, if requests within a high priority group are received while processing a low priority group, the low priority is stopped, and the memory controller will retain information concerning where processing should resume when returning to the lower priority group. This must be

*In the steps indicated with an asterisk, the switches will normally be found in the proper position for this procedure and require only a cursory check. If it is desired to clear only a small portion of memory, the INCREMENT UPPER and LOWER LIMITS may be accordingly set.

considered when connecting equipment groups to specific ports on the memory controller. The operation within an anti-hog group is such that no port may have two consecutive accesses to memory after another port has requested access. In the case of continuous access requests from all of the ports in a common anti-hog group, access is granted in cyclic order and will start with the port having the highest normal priority.

Group	Devices
1	Extended Memory Module (normally available only on GE-645)
2	Real-Time I/O Controller (custom product found only on a limited number of GE-625/635)
3	I/O Controller (including GIOC)
4 (lowest priority)	9SA Simulator Aid and Processor Modules (NOTE: Within this group, the 9SA must have the higher priority.)

Figure 4. Anti-Hog Grouping of Devices Listed in Order of Priority

As indicated above, the ANTI-HOG SWITCHES permit linking memory controller ports having link groups of devices connected so that equal access of memory may be attained. The ANTI-HOG SWITCHES and the ports they link are listed in Figure 5.

ANTI-HOG SWITCH	Ports Linked
1	0 and 1
2	1 and 2
3	2 and 3
4	3 and 4
5	4 and 5
6	5 and 6
7	6 and 7
8	Not Used; leave in middle position

Figure 5. Anti-Hog Switches

SETUP PROCEDURE FOR UNIPROCESSING GECOS

The following steps should be initiated in setting up a uniprocessing GECOS system. Since the setup procedure also depends upon the hardware configuration, the following steps would be done for a uniprocessing system consisting of the following hardware modules:

1 Processor
1 IOC
1 Memory Controller

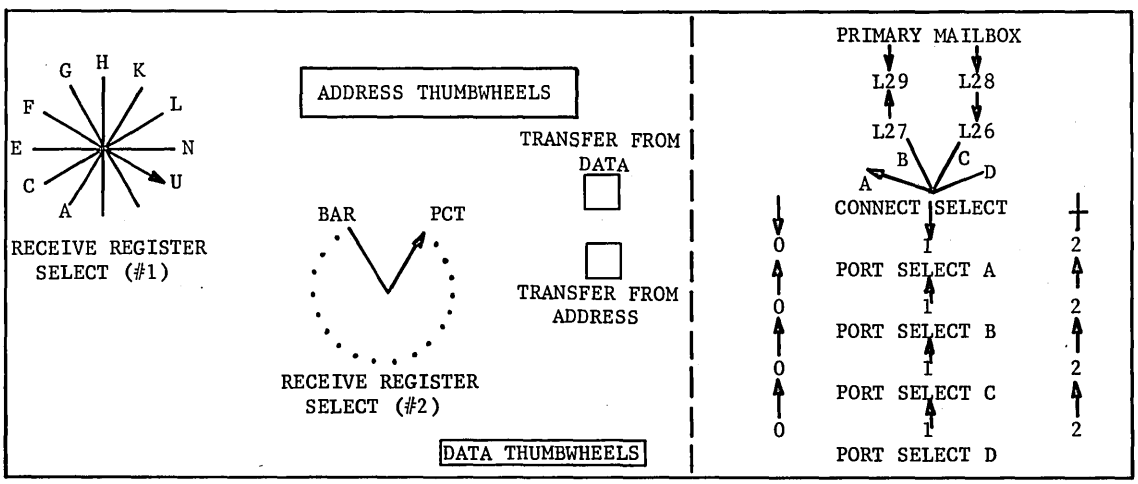
with a control memory organization as shown in Figure 6.

Address	
000000	INTERRUPT VECTORS (64 Words)
000100	GECOS COMMUNICATION BLOCK (448 Words)
001000	IOC MAILBOXES (256 Words)
001400	FAULT VECTORS (32 Words)
001440	(Beginning of GECOS Permanent Modules)

Note: If a second IOC were present, the mailboxes for this IOC would begin 256 decimal locations past the first IOC. All following addresses would be adjusted accordingly.

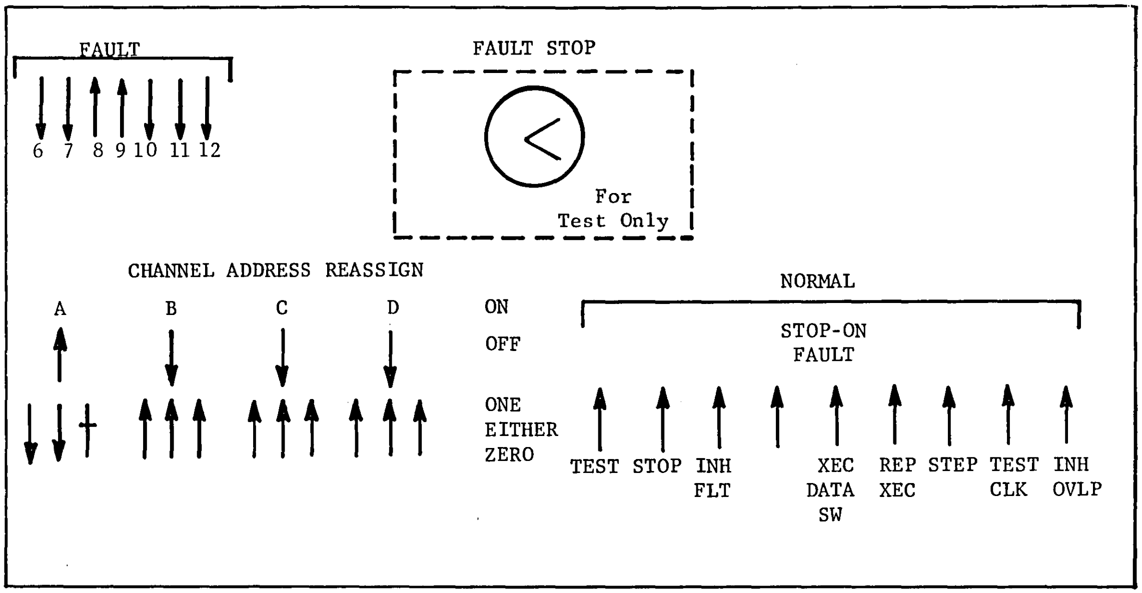
Figure 6. Examples of GECOS-III Control Memory Organization with One Processor and One IOC System

The switch settings in Figure 7 are those which would be used for uniprocessing with the hardware configuration cabled as shown in Figure 8.

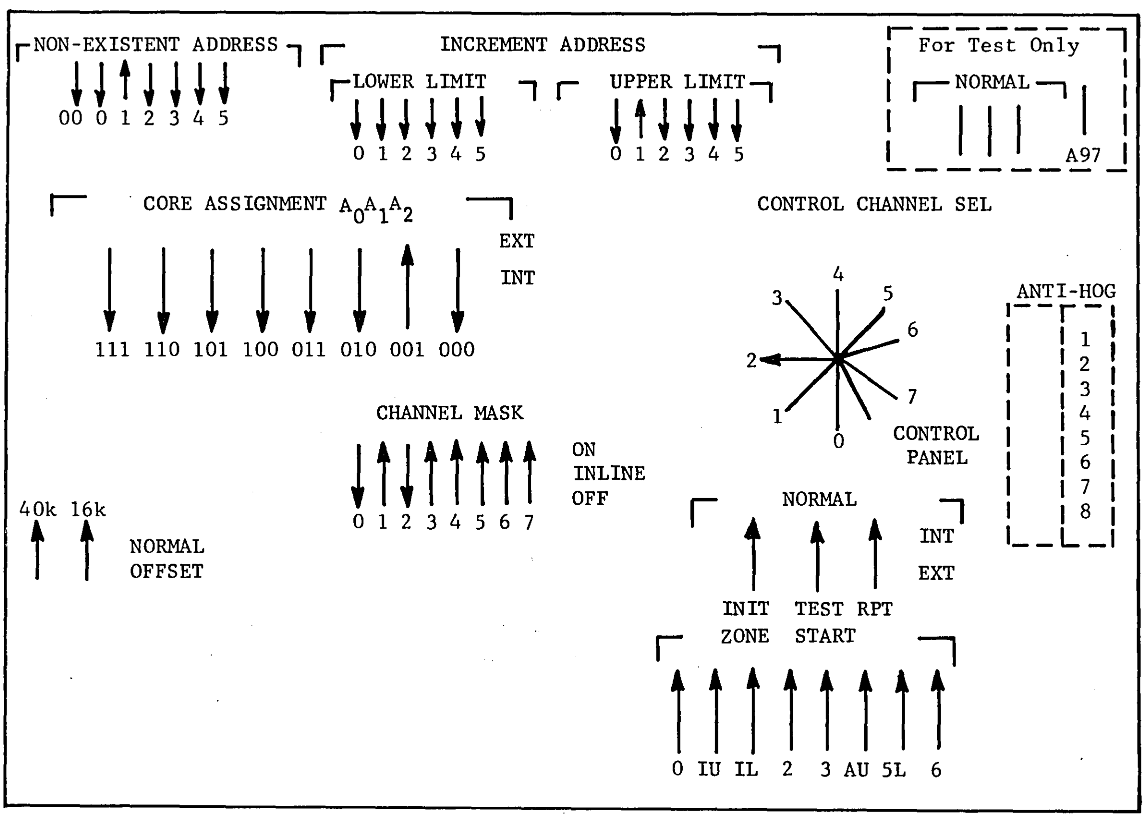


IOC T&O PANEL

Figure 7. IOC, Processor, and Memory Controller Switch Settings for Uniprocessing



PROCESSOR T&O PANEL



MEMORY CONTROLLER T&O PANEL

Figure 7. (continued)

CPB-1489

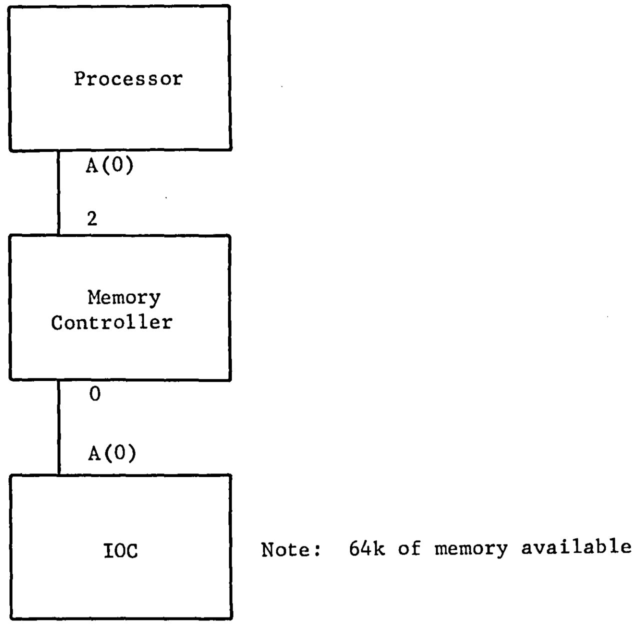
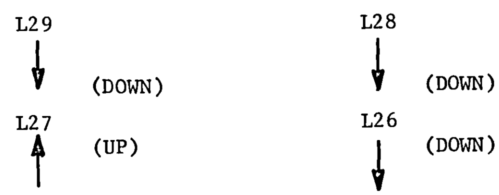


Figure 8. Cabling Diagram for a One Processor, One IOC System

IOC T&O Panel

1. Set the PRIMARY MAILBOX switches to locate the IOC Mailboxes. Switches L29 - L26 correspond to assigned memory address bits A_6, A_7, A_8, and A_9 respectively, of a 12-bit (4k) address. Since the low-order 8 bits are assumed to be zero, the IOC Mailboxes may be assigned to any address which is an integer multiple of 256 within the first 4k. The GECOS-III Control Memory Organization is depicted in Figure 6 for one IOC. The figure shows the IOC Mailboxes beginning at 1000 octal. The switch settings for the configuration shown in Figure 8 are as follows:



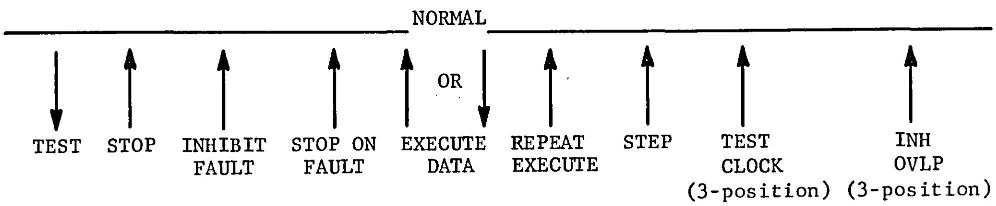
NOTE: UP = ON = 1
DOWN = OFF = 0

2. Set the CONNECT SELECT switch to A since control memory is connected to IOC-0, the only IOC present. (If control memory were connected to IOC port B, the CONNECT SELECT switch would, of course, be set to B.)

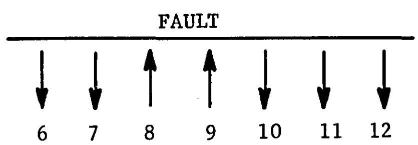
3. Set the PORT SELECT A switch to OOE (down, down, center) the PORT SELECT B to 111, and PORT SELECT PORT C and D to 111 (OFF), since only one IOC is present.

Processor T & O Panel

4. Set the Operation Control Switches as follows:



5. Set the FAULT switches to locate the Fault Vectors. The seven FAULT switches represent the 7 high-order bits of a 12-bit (4k) address. Since the 5 low-order bits are assumed to be 0, the Fault Vectors may be assigned to any address which is an integer multiple of 32 within the first 4k. The GECOS control memory organization depicted in Figure 6 for one IOC and one processor shows the fault vectors beginning at 1400 octal. The switch settings for this configuration are as follows:

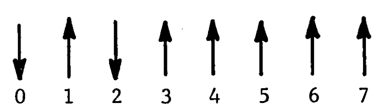


6. Return TEST switch to NORMAL.

Memory Controller T & O Panel

7. Set the ZONE switches to the UP position.

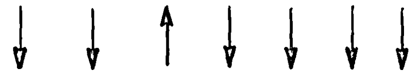
8. Set the CHANNEL MASK switches as follows:



This setup will allow memory to communicate to the IOC on port 0 and the processor to communicate on port 2, assuming the configuration shown previously in Figure 8.

9. The ANTI-HOG switches have no meaning in the example configuration (Figure 8) since this configuration does not contain "like devices." An example of the use of these switches is discussed in the preceding section, "Memory Controller Switches."

10. The NON-EXISTENT ADDRESS switches must be set to indicate that an out-of-bounds condition exists with the address of 200000 :



00 0 1 2 3 4 5

(256k)(128k)(64k)(32k)(16k)(8k)(4k)

11. Set all CORE ASSIGNMENT SWITCHES to external (EXT) except switch 001 which should be set to internal to indicate that the second core stack, making up the total 64k of memory available, is external to this memory controller, and memory addresses with the three most significant bits set to other than zero will be routed to this external core stack. (Those addresses with the bits set to zero will be routed to the internal memory stack.)

12. Set all DATA switches to ZERO (down).

13. Set DATA switches 5 and 13 UP. These switches correspond to the Initiate and Terminate Bits for the IOC in the lower half of the mask.

14. Set CONTROL CHANNEL SELECT knob to the processor channel The processor may be connected to any channel. The knob should, therefore, be set to that channel number. In the case of the configuration given in Figure 8, the CONTROL CHANNEL SELECT knob should be set to 2.

15. Set the TEST switch in the NORMAL position.

SWITCH SETTINGS FOR MULTIPROCESSING

The setup procedure for multiprocessing is given in the following paragraphs. Prior to discussing the actual procedure, some additional factors in running GECOS in a multiprocessor environment will be discussed.

The software will handle any combination up to the following:

4 Processors
4 IOC
4 Memory Controllers (256k)
The normal complement of peripherals that may be attached to the IOC's.

In any combination, two rules apply: (1) the software must be assembled to handle the combination and (2) any subset of the combinations may be run under the assembly with a redefinition of the Startup deck. (See Chapter 2.)

Once started, multiprocessing GECOS differs little from standard uniprocessing GECOS. One prime item must be kept in mind. To derive any benefit from an additional processor, the system must be pushed to its maximum. Before attempting to boot, it is imperative that all processors are initialized and are in a DIS instruction (616 op code on processor display panel. The "PA" flag must be off). Processors are initialized manually by toggling the STOP switch on the processor panel. During the startup process, only one processor, the control processor, is in operation. The control processor may be defined as the processor associated with the main typewriter console. More specifically, it is the processor that will respond to all interrupts and execute the dump for any master mode faults. Additionally, it is also the processor used in determining auto-processor shutdown and restart. As such, it is the only processor that will run continually. The control processor is determined by switch action at the memory controller(s) and by additional control cards in the Startup deck. Both of these functions will be described later.

Once the startup process is completed, initial GECOS entry is made to the Dispatcher where a connect instruction is issued to all non-control processors. Each non-control processor responds to the connect by executing coding designed to get itself started and become part of the system. By the time the first GECOS typeout is received, all processors should have been started and automatically stopped. From this point, all non-control processors will start and shut down, performing normal software functions as determined by load requirements. This is to say, any processor is capable of starting and executing any required functions (read cards, print, operate in slave mode, etc.) and no effort is made to limit any function to any processor (except with the 9SA) with the exception that only the control processor will answer interrupts and continue to operate in a light-load environment.

It is possible for a non-control processor to be started incorrectly and/or be in a fault loop, unable to do anything. If a manual restart is attempted from this condition, it is important that all non-control processors be stopped and initialized manually. As it is normal for any processor to perform all required functions (except interrupt servicing and shutdown determination), it is normal for any processor to unearth a master mode fault condition. In this instance, the control processor must be informed by the processor finding the fault, as it will be the one actually taking the dump. This notification is made by issuing a connect to the control processor. Upon receipt of this connect, the control processor will execute logic designed to halt the non-control processors. It does this by executing a connect instruction with a software "disaster" cell set. Upon responding to the connect instruction, the non-control processor(s) will place itself in a DIS condition with bit 28 set to 1, unless that processor was in an auto-stop condition. (In this case, it is a plain DIS.) The control processor will then commence the master mode dump. Indication is made on the dump and the typewriter as to which processor initially found the fault condition.

There are many ways in which a multiprocessor system may be configured physically. Because it is impossible to define each and every combination, a representative system, shown in Figure 9, is used.

The switch settings for the two processors, three memory controllers, and two IOC's are shown in Figure 10. In the case of multiprocessor systems, the ANTI-HOG switches are of importance and the switch settings for the referenced system is discussed in the following. A memory layout for the system shown in Figure 9 is given in Figure 11.

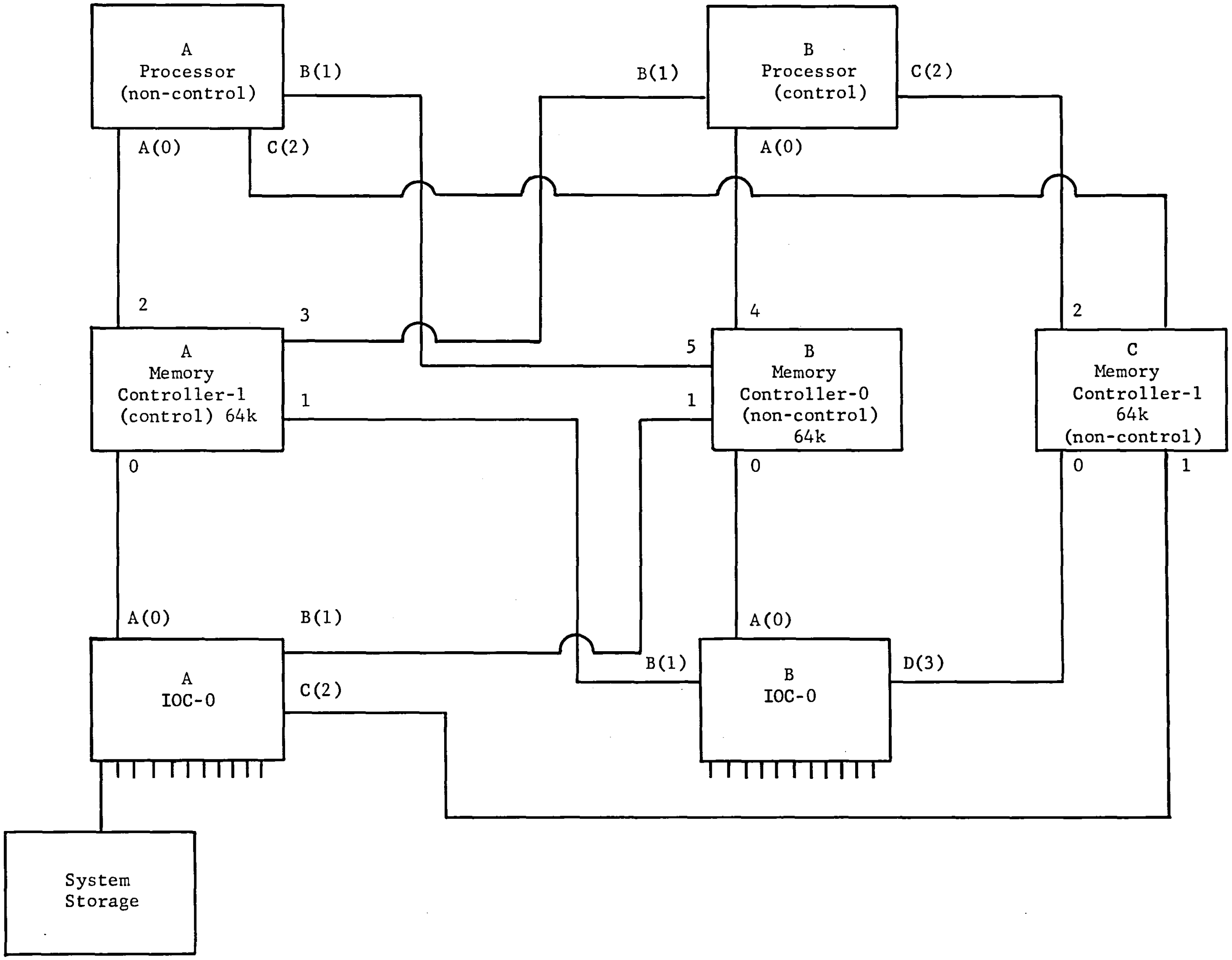
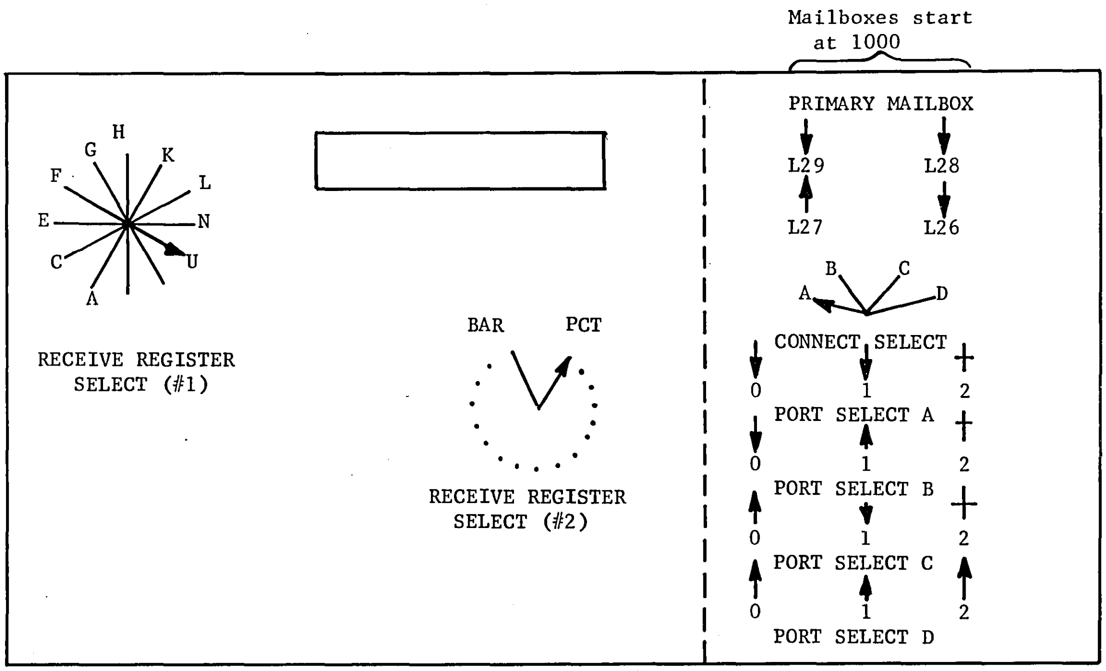
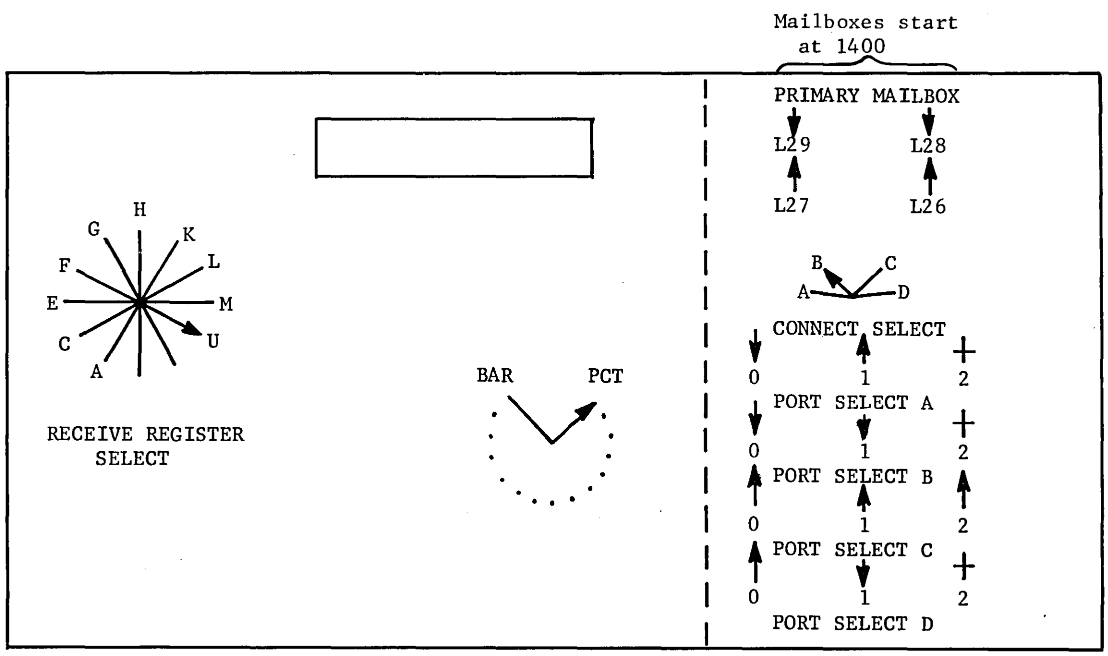


Figure 9. Cable Connectors for a Multiprocessing System



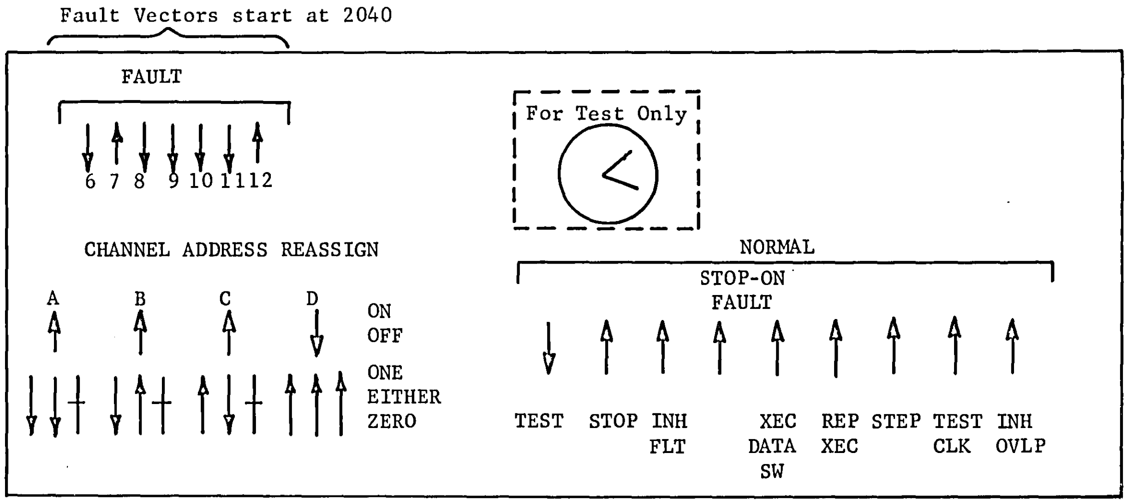
Control IOC T&O PANEL



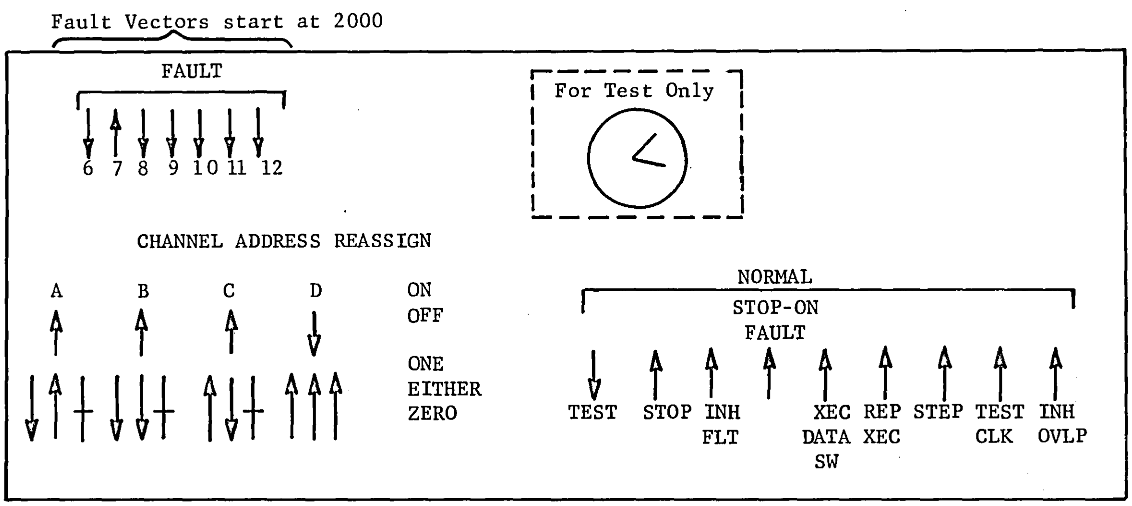
Non-Control IOC T&O PANEL

Figure 10 (a). Control and Non-control IOC Switch Settings for Multiprocessor

CPB-1489

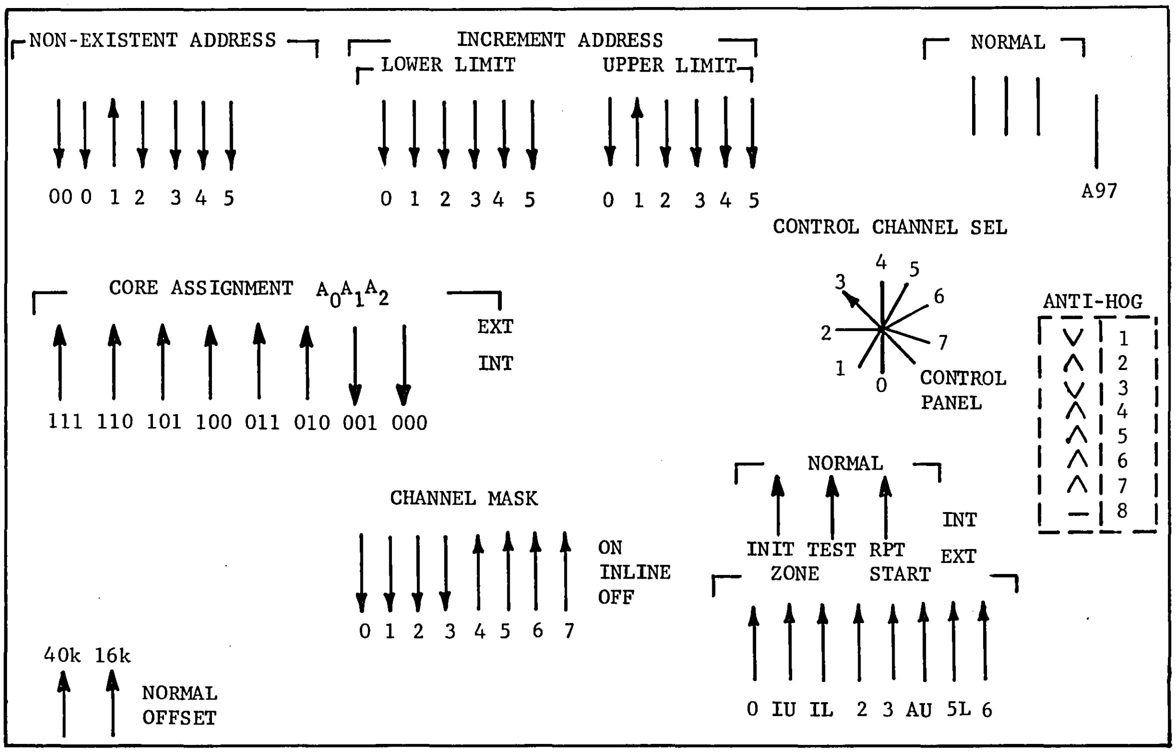


Non-Control PROCESSOR T&O PANEL

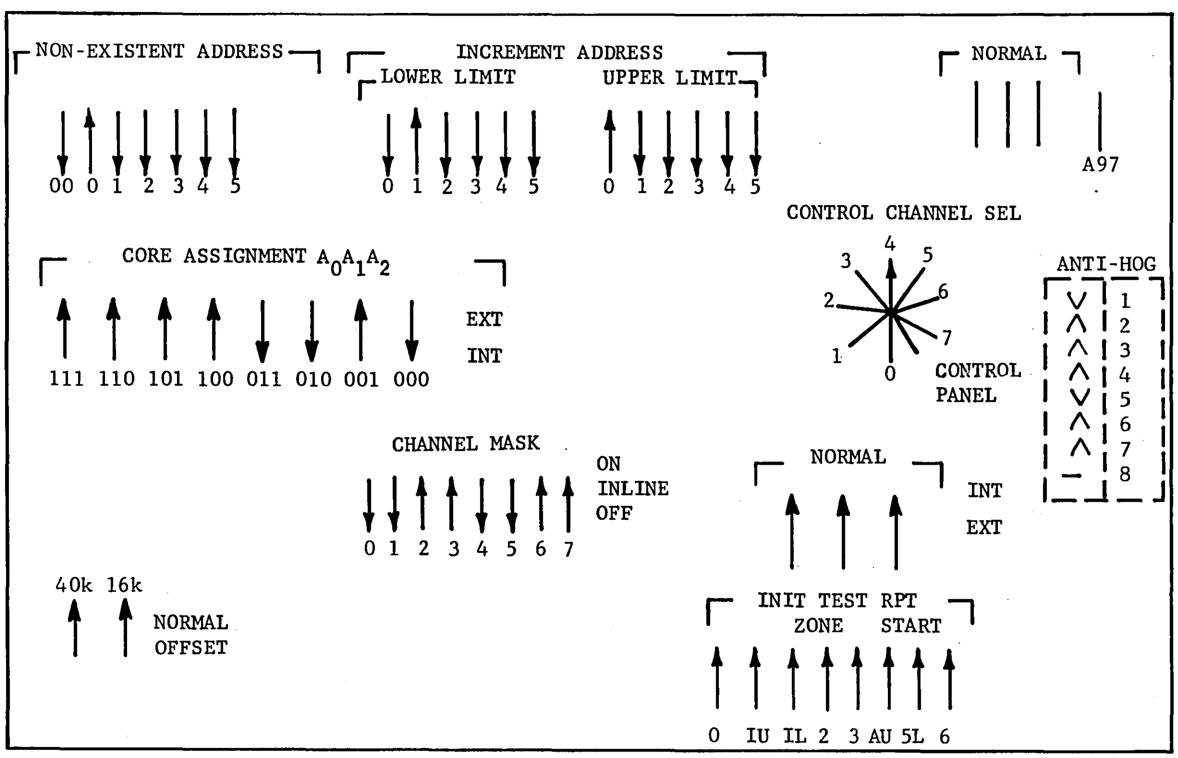


Control PROCESSOR T&O PANEL

Figure 10 (b). Non-Control and Control Processor Switch Settings for Multiprocessor

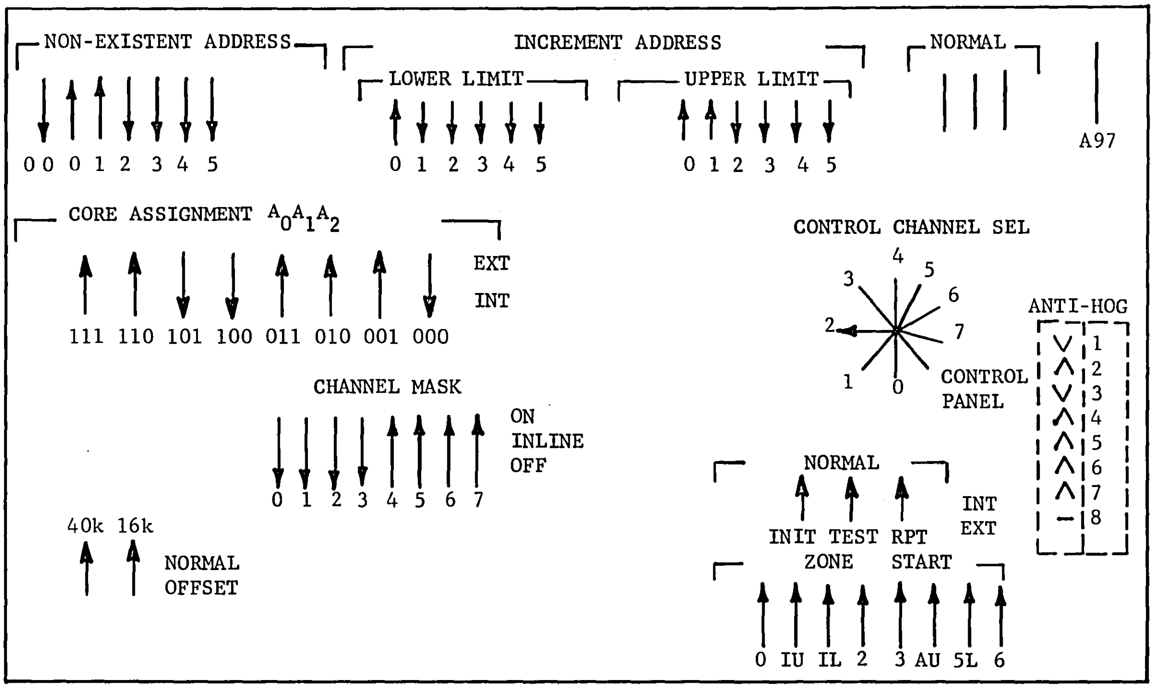


Control Memory Controller (0-64k) T&O Panel



Memory Controller (64-128k) T&O Panel

Figure 10 (c). Memory Controller Switch Settings for Multiprocessor



Memory Controller (128-192k) T&O Panel

Figure 10 (c). Continued

CPB-1489

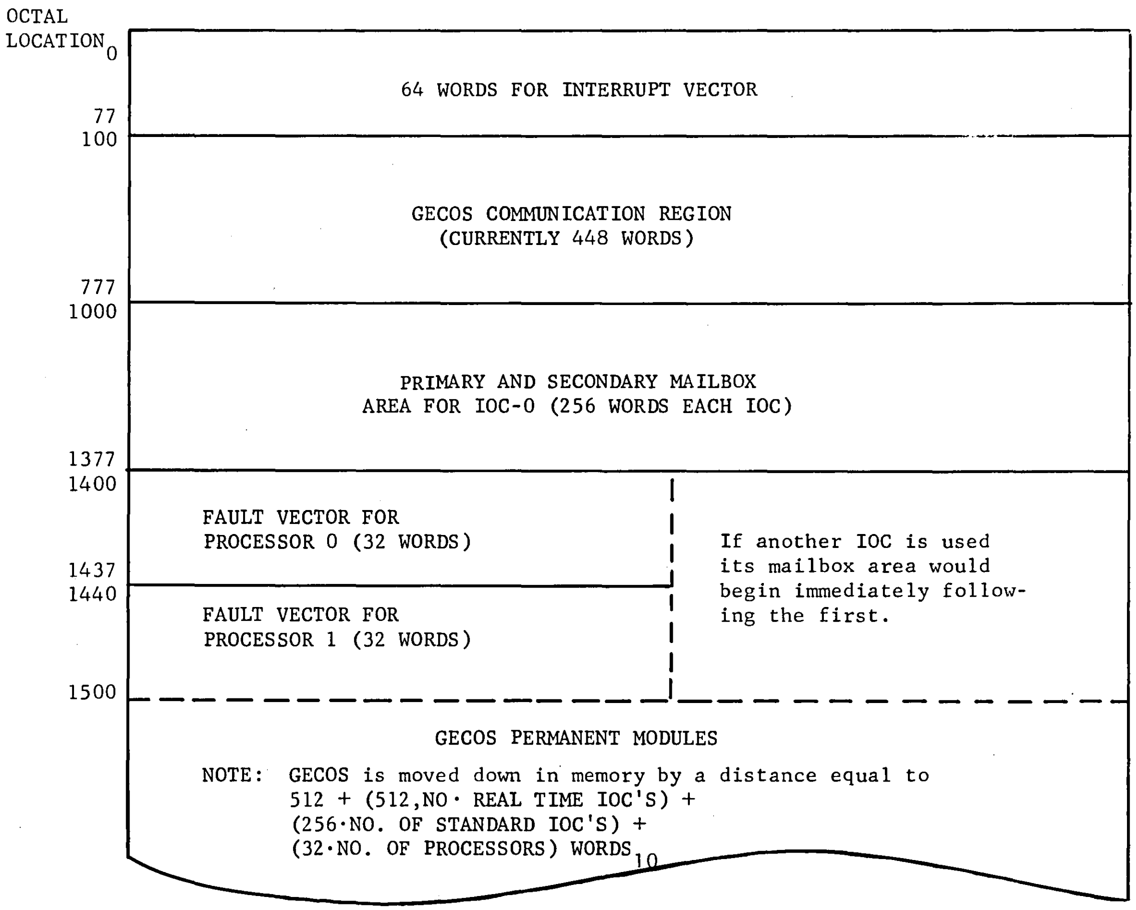


Figure 11. GECOS Lower Control Memory Layout

The ANTI-HOG switch settings are as follows:

Switch	Position
1	Down
2	Up
3	Down
4	Up
5	Up
6	Up
7	Up
8	Must be in MIDDLE position

BOOTSTRAPPING THE STARTUP DECK

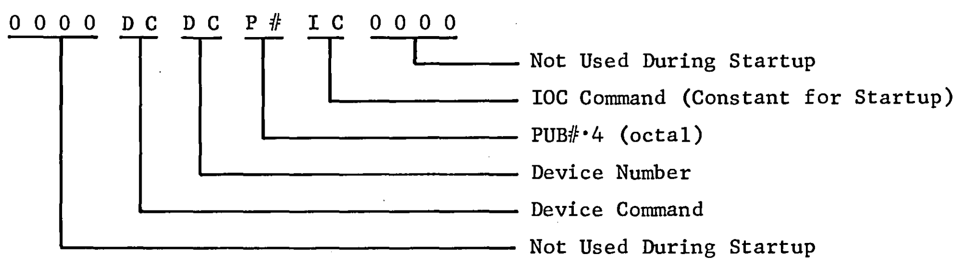
The actual bootstrap procedure does not differ between Uniprocessing and Multiprocessing GECOS, however the Startup deck does differ slightly. The procedure below should be followed after the setup procedure discussed previously.

At the Card Reader:

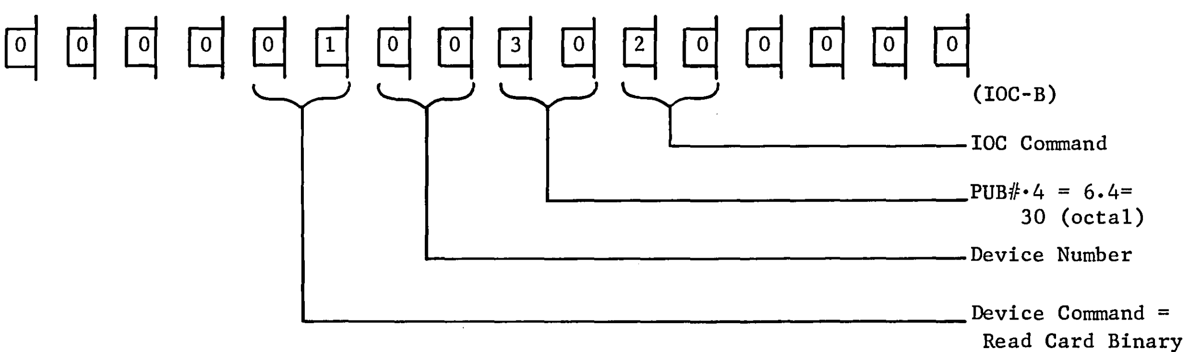
Put the STRT deck into the hopper and press OPERATE/RESET so that the OPERATE indicator comes on.

At the IOC T&O Panel:

1. If RUN light is on, press RUN to turn it off.
2. Set the ADDRESS thumbwheels to 0050 on an IOC-B or to 02 on an IOC-C.
3. Enter the Device Command, Device Number, PUB (Channel) Number multiplied by 4 (octal), and IOC Command in the DATA thumbwheels on an IOC-B or in the DATA-IN toggle switches on an IOC-C. The format of the data to be entered is as follows:



The specific setting for the DATA thumbwheels for a Card Bootstrap for GE-625/635 system with the Card Reader on PUB #6 as follows:



4. Turn REGISTER SELECT (#1; see Figure 10) to U position.

5. Turn REGISTER SELECT (#2; see Figure 10) to PCT position.

On the Console:

6. Depress INIT Button.

7. Depress the BOOTLOAD Button.

8. Repeat steps 2 and 3 to continue reading cards.

After the second card is read, memory is searched for the location of the Terminate Interrupt. This determines the IOC Number, Card Reader PUB, and the Card Reader Mailbox Address. With this information, the Bootstrap builds a card read routine and loads Startup which, in turn, loads the GECOS System.

2. STARTUP DECK

The Startup Deck consists of the Startup Program, System Description Cards, and any GECOS correction modules required. The Startup program provides initialization and utility functions needed to establish GECOS on a variety of machine configurations. The cards are arranged as follows:

STARTUP PROGRAM

SYSTEM DESCRIPTION CARDS

$CONFIG	(Hardware configuration description section)
$EDIT	(File edit control section)
$FILES	(Software configuration description section)
$PATCH	(System program patch control section)
$LOAD	(GECOS correction modules section)

Each of the sections must begin with a $ section card (for example, $CONFIG or $EDIT) and end with an ***EOF card.

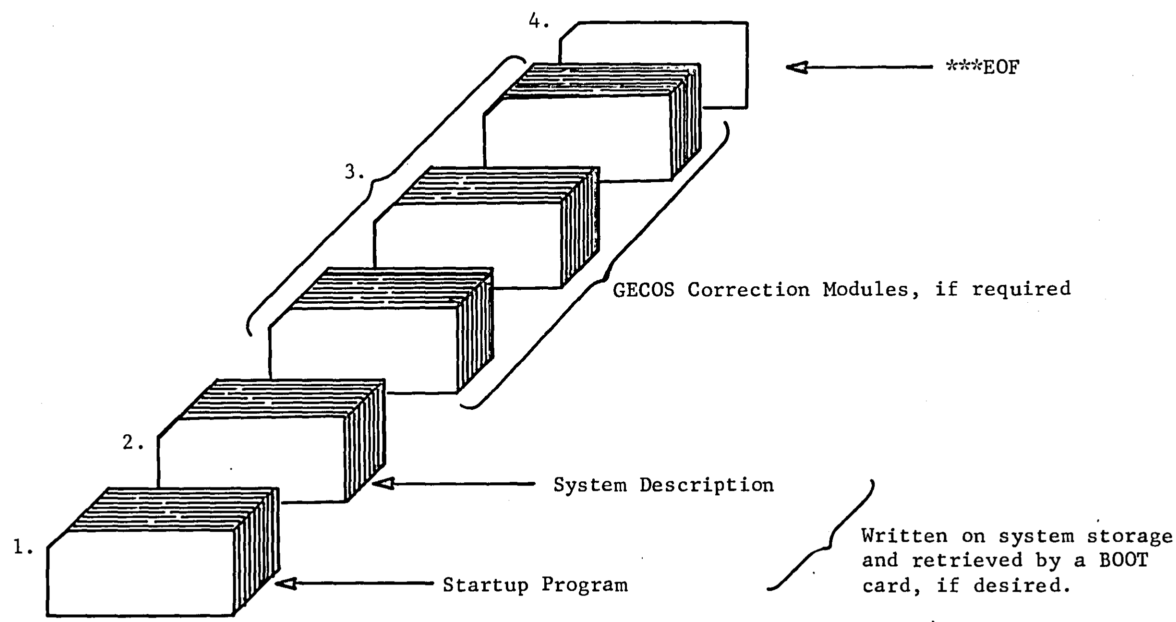


Figure 12. Startup Deck Setup

The Startup Program is loaded into the upper end of the first 64k of memory (currently 140000 octal) via the Startup Program card deck. The Startup Program first zeros memory below and above itself; it then sets the Interrupt Mask, the Interrupt Vectors, the IOC Mailboxes, and the Fault Vector as required for Startup operation. Startup then reads Packet 2, the System Description Cards, and builds a set of System Description Tables within itself.

The System Description cards are entered as Packet 2 of the GECOS System Startup Deck. The System Description cards perform two functions:

- Definition of the GE-625/635 System to be controlled by GECOS*
- Redefinition of the peripherals assigned to perform system functions (for example, System Storage, SYSOUT, etc.).

If these cards are incorrectly prepared, error messages, described in CPB-1477, will be issued.

During system initialization, each permanent module may examine the System Description Tables (described in the following section) which were created by Startup from the System Description cards. Each module modifies itself to control the exact system configuration defined and may request that other modules be loaded if necessary. If, for any reason, the defined configuration requires modification (reassignment of one or more system functions, deletion of a channel or device which requires maintenance, etc.), this may be readily accomplished by reinitializing GECOS with the System Description cards altered to reflect the desired changes.

The System Description cards consist of a variable number of cards which define:

1. Peripheral devices used by Startup
2. System storage devices
3. IOC Peripheral Unit Buffer (PUB) assignments
4. Memory port assignments of each memory controller
5. Size of core memory
6. Magnetic tape channels which are crossbarred
7. Number of magnetic tape devices
8. Number of disc storage devices
9. Processor for the Simulator Aid 7090/7094 (referred to as 9SA device)
10. Control processor, if applicable
11. SYSOUT media
12. Location, on system storage, of the master catalog and the GECOS system

*The system defined may never be more than the maximum configuration for which GECOS was assembled or the maximum hardware configuration of the specific system GECOS is to control; but it may define a lesser configuration if desired and certain GECOS system functions must be on IOC-0.

The following System Description Cards are discussed in this section:

```
$CONFIG
   $ SYID
   $ DATE
   $ TRACE
   $ MCT
   $ IOC
   $ XBAR
   $ 9SA

$EDIT
   $ INIT
   $ FILDEF
   $ PERMCP
   $ DUMP

$FILES
   $ SYSTEM
   $ SAVE
   $ SYSOUT
   $ LIBRARY
   $ ACCOUNT

$PATCH
   OCTAL

$LOAD
   $ OBJECT
     Module deck
   $ DKEND
```

Each System Description Card must have a $ in column 1, blanks in column 2-7, the card type in column 8-14, and the variable field beginning in column 16. The first blank in the variable field terminates the scan. The first card of a System Description deck must be a $CONFIG card, The last card of each section must be an ***EOF card. All $ ETC cards used to continue the variable field must immediately follow the appropriate $ xxxx card.

$CONFIG SECTION

The hardware configuration to be used is defined in the $CONFIG section. The control cards define all device connections in the system, all device characteristics, and the maximum machine configuration. DATE and TRACE control cards are also assigned to this section.

Internal configuration and device control tables are derived from $CONFIG information. A roll call is executed at the time GECOS is initialized to verify the connections and determine status of peripherals. Devices which are inoperable at Startup are released from consideration but the entries remain on the internal tables so the devices may be returned to the system at a later time.

DATE

General Form:

1	8	16
$	DATE	mmddyy

The entry for the date consists of six numeric characters where mm is the month, dd the day, and yy the year.

Examples:

```
1      8       16
$      DATE    051567
$      DATE    120267
```

SYID (System Identification)

General Form:

```
1      8       16
$      SYID    xxxxxx
```

The entry for system identification consists of six alphanumeric characters.

Examples:

```
1      8       16
$      SYID    SN09
$      SYID    C+FGC2
```

TRACE

General Form:

```
1      8       16
$      TRACE   control value [,control value,abort entry]
```

The entry for system trace control specifies which of the system trace entries are to be included in the internal system trace and which trace entry is to cause a system abort.

The control values are octal numbers representing the bits of two trace control words. These octal values are converted to right-justified binary form for internal use. The bits of the two control words are assumed to be numbered from left to right in the normal fashion. Each numbered bit value of zero indicates that the trace entry is to be included in the table. A value of one indicates that the trace entry is not to be included.

The abort entry number is also an octal value which specifies the number of the internal trace table entry which is to cause a system abort. This option is effective only when a partial trace is specified by the two preceding control values.

Examples:

```
1      8       16
$      TRACE   007777777777
$      TRACE   767777777777,777777777777,20
```

MCT (Memory Controller)

General Form:

```
1      8       16
$      |MCT-#  |size,PORT-#,module,PORT-#,module
$      |ETC    |[PORT-#,module] ...
```

The entry for a memory controller specifies the memory controller number (0-3), size of attached memory (1-128, where the value represents the number of 1024-word modules), and the hardware module attached to the memory controller ports (0-7).

The module type codes listed below are the allowable entries in the module fields. The type codes, other than 9SA, require a qualifying number from 0-3 following the dash.

MODULE TYPE CODES

MODULE CODE	MODULE DESCRIPTION
IOC-#	Input/output controller, models A and B
IOCC-#	Input/output controller, model C
RIOC-#	Real-time input/output controller
PRO-#	Processor
9SA	Simulation aid

There are some restrictions imposed upon allowable input/output controller combinations within a hardware system to be controlled by GECOS. These restrictions are imposed by Startup when the memory controller entries of the hardware configuration section are processed. The most important consideration is that IOC models A and B may not exist within the same configuration as an IOC model C. In addition, GECOS will accommodate only the precise number of real-time IOC's provided for when the GECOS modules are converted from symbolic to binary form.

The optional field groups of the MCT entry may be repeated to define up to eight memory controller port connections.

Examples:

```
1      8       16
$      |MCT-0  |64,PORT-0,IOC-0,PORT-2,PRO-0
$      |MCT-1  |32,PORT-0,RIOC-0,PORT-1,IOC-0,
$      |ETC    |PORT-3,PRO-0,PORT-5,9SA
$      |MCT-0  |64,PORT-0,IOCC-0,PORT-1,PRO-0
$      |MCT-1  |64,PORT-0,IOC-0,PORT-1,9SA
$      |ETC    |PORT-2,PRO-0,PORT-3, PRO-1
```

IOC Channel

General Form:

```
1        8            16
$        {IOC }-#      PUB-#,device type[,UNITS-#]
         {IOCC}
$        ETC          [[,UNIT-#][,device name]  ...
$        ETC          [[,ASA7     ]
                       [,ASA9     ]    [,dedicated]] ...
                       [,MODULES-#]
```

The CONFIG entry for an IOC channel specifies the IOC number (0-3), channel (PUB) number (0-15), and the type of peripheral device attached. The IOC model, number of attached units, and individual unit characteristics may also be included.

The IOC model is used for documentation purposes only. The model type has been previously determined by entries on the MCT cards.

The allowable entries in the device-type field are given below:

DEVICE-TYPE CODE	DEVICE DESCRIPTION
DISC*250	Large disc (DSU270)
DISC*200	Standard disc (DSU200)
DISC*150	Removable disc (DSU160)
DRUM*200	Drum (MDU200)
DRUM*300	Drum (MDU300)
RACE	Magnetic card (MSU388)
TAPE	Standard Tape
TAPE*ASA7	ASA compatible tape (7-track)
TAPE*ASA9	ASA compatible tape (9-track)
READER	Card reader (CRZ200)
READER*200	Dual-stacker card reader (CRZ201)
PUNCH*100	Card punch (CPZ100)
PUNCH*200	Card punch (CPZ200)
PUNCH*300	Card punch (CPZ201)
PRINTER	Line printer (PRT201)
PTAPE	Paper tape reader-punch (PTS200)
CONSOLE	Console
REMOTE	Remote processor (DATANET-30)

The number of units connected to a channel may be specified in the units field. This value is limited by the type of attached device, and the maximum possible number of units is assumed to be present if the units field is absent. Maximum values for all types of devices are given on the following page.

The optional field groups may be repeated to define the characteristics of all units attached to the channel.

A unit number may be specified in the unit field. Unit description fields which follow always apply to the last previous unit number. The unit number is limited by the number of units specified or assumed for the channel and the address range for the device type. The address ranges for all types of devices are given on the following page. If the unit field is not present, the following unit description fields are assumed to apply to the lowest addressable unit for the device type.

One or more device names may be associated with a unit through use of device name fields. A device name consists of three alphanumeric characters, the second of which is always alphabetic.

The number of recording tracks on an ASA-compatible magnetic tape unit may be specified by use of an ASA7 or ASA9 field when types of handlers are mixed. The ASA7 entry indicates seven track recording and the ASA9 entry indicates nine track recording. Also the number of storage modules contained in a random access unit may be specified by use of a modules field. This number has a value and the maximum possible number of modules are assumed to be present if the modules field is absent. Maximum values for all types of devices are given below.

A unit may be dedicated to use only by those programs which specifically call for it by name. This is specified through inclusion of the dedication field.

DEVICE CHARACTERISTICS

DEVICE TYPE CODE	MAXIMUM NO. UNITS(n)	LOW UNIT ADDRESS	HIGH UNIT ADDRESS	MAXIMUM NO. MODULES
DISC*250	1	1	1	4
DISC*200	4	1	4	4
DISC*150	10	1	10	
DRUM*200	1	0	0	2
DRUM*300	2	0	1	
RACE	2	0	1	
TAPE	16	0	15	
TAPE*ASA7	16	0	15	
TAPE*ASA9	16	0	15	
READER	1	0	0	
READER*200	1	0	0	
PUNCH*100	1	0	0	
PUNCH*200	1	0	0	
PUNCH*300	1	0	0	
PRINTER	1	0	0	
PTAPE	1	0	0	
CONSOLE	1	0	0	
REMOTE	1	0	0	

Examples:

1	8	16
$	IOC-0	PUB-0,DRUM*200,ST1
$	IOC-0	PUB-1,TAPE,UNITS-8
$	IOC-0	PUB-3,TAPE,UNITS-8,UNIT-0,LIB,
$	ETC	DEDICATED
$	IOC-1	PUB-1,TAPE*ASA7,UNIT-0,MT0,UNIT-1
$	ETC	MT1,ASA9,DEDICATED,UNIT-2,MT2
$	IOCC-0	PUB-6,READER,CR1
$	IOCC-1	PUB-10,PRINTER,FTC,DEDICATED
$	IOC-2	PUB-4,DISC*200,UNITS-2,UNIT-1,
$	ETC	DS1,MODULES-2,DEDICATED,UNIT-2,DS2
$	IOC-0	PUB-8,CONSOLE,TY1,TY2,TY3,TY4
$	IOC-1	PUB-15,PUNCH*300,UNITS-1,PU1
$	IOC-0	PUB-4,DISC*200,UNITS-1,ST1,DS1

XBAR (Crossbar Arrangement)

General Form:

1	8	16
$	XBAR	$\left\{\begin{matrix}\text{IOC}\\\text{IOCC}\end{matrix}\right\}\text{-\#,PUB-\#,}\left\{\begin{matrix}\text{PUB-\#}\\\left\{\begin{matrix}\text{IOC}\\\text{IOCC}\end{matrix}\right\}\text{-\#,PUB-\#}\end{matrix}\right\}$
$	ETC	$\left[\text{,}\left\{\begin{matrix}\text{PUB-\#}\\\left\{\begin{matrix}\text{IOC}\\\text{IOCC}\end{matrix}\right\}\text{-\#,PUB-\#}\end{matrix}\right\}\right]\ldots$

The hardware configuration entry for a crossbar arrangement specifies all of the IOC channels which are mutually crossbarred.

The IOC model is indicated for documentation purposes only. The IOC model type for the hardware system is determined by the entries on the memory controller cards.

The IOC numbers may be 0-3. The PUB numbers may be 0-15. The PUB number entries which follow an IOC field always apply to the last previous IOC number.

Examples:

1	8	16
$	XBAR	IOC-0,PUB-1,PUB-2
$	XBAR	IOC-1,PUB-3,PUB-5
$	XBAR	IOC-0,PUB-1,PUB-2,IOC-1,PUB-1,PUB-2
$	XBAR	IOC-0,PUB-1,IOC-1,PUB-2
$	XBAR	IOCC-0,PUB-3,PUB-5

9SA (Simulation Aid)

General form:

```
      1      8       16
     |$      |9SA    |PRO-#
```

The configuration entry for a simulation aid specifies the number of the processor which is attached to the simulation aid. This number may be 0-3.

Example:

```
      1      8       16
     |$      |9SA    |PRO-0
     |$      |9SA    |PRO-1
```

Restrictions

CARD ORDER

The MCT card which describes memory controller number zero must precede all IOC, IOCC, and XBAR cards within the hardware configuration description. This MCT card is used to determine the number of input/output controllers, real-time input/output controllers, and processors in the system. These values determine the origin for the configuration tables derived from the IOC, IOCC, and XBAR cards.

A XBAR card must follow the IOC or IOCC card which describes the primary, or first, channel entered on the XBAR card. The configuration table entries for the non-primary channels are derived from those for the primary channel when the XBAR card is processed.

CONFIGURATION DUPLICATION

IOC or IOCC cards should not be included in the hardware configuration description for non-primary crossbarred channels. The configuration table entries for the non-primary channels are derived from those for the primary channel when the XBAR card is processed.

DEVICE NAMES

A random access device with the name ST1 must be included in the configuration deck. This device is considered to be the primary system storage. The configuration deck must also include entries for four console names TY1, TY2, TY3, and TY4, even when there is only one console typewriter in the system. Messages are assigned to these consoles by name.

Sample Hardware Configuration

```
$CONFIG

$    DATE          052467
$    MCT-0         64,PORT-0,IOC-0,PORT-1,PRO-0
$    IOC-0         PUB-0,DRUM*200,ST1
$    IOC-0         PUB-1,TAPE,UNITS-8
$    IOC-0         PUB-3,TAPE,UNITS-8,UNIT-3,MT3,
$    ETC           UNIT-4,MT4
$    IOC-0         PUB-4,DISC*200,UNITS-8,UNIT-3,MT3,
$    ETC           DS1,UNIT-2,DS2
$    IOC-0         PUB-6,READER
$    IOC-0         PUB-8,CONSOLE,TY1,TY2,TY3,TY4
$    IOC-0         PUB-10,PRINTER
$    IOC-0         PUB-13,REMOTE
$    IOC-0         PUB-14,PUNCH*200
$    XBAR          IOC-0,PUB-1,PUB-2
$    XBAR          IOC-0,PUB-3,PUB-5
***EOF
```

$EDIT (FILE EDIT SECTION)

Random access file edit functions needed to establish the GECOS and software systems on the machine are defined by entries in the Startup file edit control section. These entries provide information needed for file system catalog creation and file copy control.

The Startup file edit functions are not intended to provide the full range of capabilities needed for software system maintenance. Startup provides the functions needed to establish system files on a bare machine. Extensions and alterations to this basic system may be created through use of the system editor after a GECOS system has been placed into operation.

The file edit control section may contain one entry for each of the following items:

1. Catalog initialization
2. System file copy
3. Data file copy
4. Random file definition
5. Non-random file definition
6. Catalog position
7. Random file dump

INIT (Catalog Initialization)

General Form:

```
1       8        16
$       INIT     device name [,device name]  ....
```

The file edit control entry for catalog initialization specifies the logical device names for the random devices to be initialized.

The device name consists of three alphanumeric characters, the second of which must be alphabetic. This name must be associated with a physical device by an entry within the configuration description.

The INIT entry causes the file system master catalogs to be initialized and LINK and LLINK storage control tables to be created and written on the storage device.

Examples:

```
1       8        16
$       INIT     ST1,DS1,DS2
$       INIT     DS1
```

FILDEF (File Copy and Definition)

General Form:

```
1       8        16
$       FILDEF   device name,catalog name/file name,
                         size [/Ø ] [,format,tape device]
```

The file edit control entry specifies the device on which the file is to be defined, file identification, size of the file and, when necessary, the format of the file being copied and tape unit on which it is located.

The device name is an alphanumeric value consisting of three characters. This name must be associated with a physical device by an entry in the configuration description.

The catalog name and file name are alphanumeric values consisting of from one to twelve characters.

The file size field is a decimal number specifying the number of links to be reserved for the file. The /Ø option is used if the entry is to override an existing file of the same name. When /Ø is not used or the file to be overridden is smaller than the file described on this card, the catalog blocks defining the file are written and the necessary links reserved but the contents of the described file are not entered into the system. A typewriter message indicates this condition.

The format field entry is either SYS or RDM. SYS indicates the file to be copied is in GECOS III system loadable format. RDM indicates a data file.

The tape device field contains the name of the tape unit from which the file is to be copied. An asterisk may be used in the tape device field when the last device defined contains the file to be copied.

Examples:

```
1       8       16
$       FILDEF  ST1,SYS/SYOU,400
$       FILDEF  DS1,GECOS3/TMSHRE,20,SYS,3T4
```

PERMCP (GECOS II File Copy)

General Form:

```
1       8       16
$       PERMCP  device name,tape device [,OMIT,filename(s),...   ]
                                        [,SELECT,filename(s),... ]
```

The GECOS II file copy entry enables files in the GECOS II PERM file format to be entered in the GECOS III system at startup time. The entry must specify the device name on which the file is to be copied and the tape device from which it is copied. The OMIT or SELECT options are used when only portions of a tape are needed.

The device name is an alphanumeric value consisting of three characters. This name must be associated with a physical device by an entry in the configuration description.

The tape device field contains the name of the tape unit from which the file is to be copied.

In the optional field entry, OMIT causes the files named to be omitted from the copy. SELECT causes only the files named to be copied. The file names in this entry do not include the catalog name.

Examples:

```
1       8       16
$       PERMCP  DS2,3T5
$       PERMCP  DS1,ET2,SELECT,COBOL,ILANG,A2
$       PERMCP  ST1,3T4,OMIT,FORTRANII,ALGOL
```

DUMP (Random File Dump)

General Form:

```
1       8       16
$       DUMP    device,{LINK } -#,size
                       {LLINK}
$       ETC     ,device name,{LINK } -#,size...
                             {LLINK}
```

The file edit control entry for a random file dump specifies the name of the random device which contains the file segment, the segment type and origin, and the size of the segment to be dumped.

The device name is an alphanumeric value which consists of three characters. This name must be associated with a physical device by an entry within the configuration description.

The file segment type field indicates the type of file entry to be dumped. The LLINK storage within a file is reserved to file system catalogs. The LINK storage contains the contents of the files themselves. The file segment origin is the number of a storage element within the type of file entry where the dump is to begin.

The size of the segment to be dumped is a decimal value which indicates the number of file segment elements to be dumped.

The file dump entry causes successive elements of the indicated type of file segment to be read and dumped on the printer until the indicated number of elements have been processed.

Examples:

1	8	16
$	DUMP	ST1,LINK-10,10
$	DUMP	DS1,LLINK

Restrictions

CARD ORDER

If a device is to be initialized, the $ INIT card for that device must precede any edit control cards defining files on that device.

FILE NAMES

All file names on the FILDEF cards are preceded by a catalog name and separated from it by a slash (/).

Sample File Edit

```
$EDIT

$     INIT       ST1
$     FILDEF     SD1,GECOS3/DISC,25/0,SYS,3T3
$     FILDEF     ST1,GECOS3/DRUM,15,SYS,3T2
$     FILDEF     ST1,GECOS3/SOFTWARE,20/0,SYS,3T1
$     FILDEF     DS1,GECOS3/DUMP,3
$     PERMCP     DS1,3T4,SELECT,RARESTUFF
***EOF
```

$FILES (SOFTWARE CONFIGURATION SECTION)

The software configuration to be established on the machine is defined by entries in the software configuration control section. These entries describe the files which contain software programs to be used and the files to be used for fixed system storage.

The software program complement available to GECOS during system operation is defined by the internal program directory. This directory is derived from the system files specified at Startup. Thus, many independent software files may be established on the random devices of the machine. Only those files indicated at Startup are configured into the software system.

The files used by GECOS for fixed system storage are defined by internal file control tables. These tables are also derived from the system file entries specified at Startup.

The software configuration control section may contain one entry for each of the following items.

1. System program files
2. System save file
3. System output files
4. System library file
5. Accounting file

SYSTEM (System Program Files)

General Form:

1	8	16
$	SYSTEM	file name[,file name] ...

The software configuration entry for system program files specifies the names of up to eight files which are to constitute the GECOS and software systems.

The file names are alphanumeric values of the form defined for the GECOS file system. This means that concatenated catalog names must precede the file name to describe any file structure created by the file system.

Examples:

1	8	16
$	SYSTEM	GECOS3/TMSHRE,GECOS3/DRUM,
$	ETC	GECOS3/DISC,
$	ETC	GECOS3/SOFTWARE

SAVE (System Input File)

General Form:

1	8	16
$	SAVE	file name

The software configuration entry for the system save file specifies the name of the file to be used by the system for dump control and restart procedures. It is also used as a pushdown file during system initialization.

The file name must conform to the same conventions as those described above for the system program files entry.

Examples:

```
1      8       16
$      SAVE    GECOS3/DUMP
$      SAVE    GECOS3/RESTART
```

SYSOUT (System Output Files)

General Form:

```
1      8       16
$      SYSOUT  file name [,file name] ...
```

The software configuration entry for system output files specifies the names of up to four files which are to be used for system output collection.

The file names must conform to the same conventions as those described above for system program files.

The system output files must all be assigned on the same type of random device. In addition, all the files must have the same size. If the files are not the same size, the size of the smallest file is assumed to be the size of all files.

Examples:

```
1      8       16
$      SYSOUT  GECOS3/SYSOUT
$      SYSOUT  GECOS3/SYSOUT1,GECOS3/SYSOUT2
```

LIBRARY (System Library File)

General Form:

```
1      8       16
$      LIBRARY { RDM,filename    }  [ ,RDM,filename    ]
               { RMV,device name }  [ ,RMV,device name ]
```

The software configuration entry for the system library file specifies the name of the file(s) to be used for the system library. Up to two files may be defined on one card.

The file name must conform to the conventions described above for system program files. If the file is on a non-random device, its location is specified instead of its name.

The system library file may be assigned to a non-random storage device. This is indicated when RMV precedes the file location.

Example:

1	8	16
$	LIBRARY	RMV,3T6
$	LIBRARY	RDM,GECOS3/LIB1,RMV,3T1

ACCOUNT (System Accounting File)

General Form:

1	8	16
$	ACCOUNT	{ RDM,filename or location RMV,device name }

The software configuration entry for the system accounting file specifies the name of the file to be used to store accounting information.

The file name must conform to the standard GECOS file system conventions. These conventions are summarized above in the system program files section.

The system accounting file may be assigned to a non-random storage device in the same way as the library file.

Examples:

1	8	16
$	ACCOUNT	RDM,GECOS3/ACCOUNT
$	ACCOUNT	RMV,3T7

Sample Software Configuration

```
$FILES

$     SYSTEM    GECOS3/DRUM,
$     ETC       GECOS3/DISC,GECOS3/SOFTWARE
$     SAVE      GECOS3/DUMP
$     SYSOUT    GECOS3/SYOT1,GECOS3/SYOT2
$     LIBRARY   RDM,GECOS3/LIB1,RMV,3T6
$     ACCOUNT   RMV,3T7
***EOF
```

$PATCH (PROGRAM PATCH SECTION)

Alterations to GECOS and software programs are defined by entries in the system program patch section. These entries describe absolute, octal alterations to be applied to specified programs when they are loaded for execution.

Patches to be applied to system programs are defined by the internal patch table. This table is derived from the alteration cards entered at Startup. The patch table is used by all GECOS routines which load programs from system storage.

PATCH

General Form:

```
address       OCTAL     value[,value,]... ,program name
```

The program patch entry specifies the load address, values to be loaded, and the name of the program to be altered.

The load address is a left-justified, octal value from one to six digits. If the address is absent, the last used load address plus one is used.

The successive values to be loaded are entered as octal numbers of from one to twelve digits. The first value is loaded at the specified or implied address. Following values are loaded at following consecutive addresses.

The program name is a left-justified, BCD value which is the same as that used in creation of the system loadable file containing the program. This name is entered in the card identification field.

Examples:

```
536      OCTAL      5710004,0221003,.MFALT
2204     OCTAL      011007,.FORT1
```

Sample Program Patch Section

```
$PATCH

2640     OCTAL      777777710204,.MDISP
2246     OCTAL      2272752070,.MALC1
614      OCTAL      3341236012,2710756014,.MBRT6
527      OCTAL      77364207,777734442204,.MRELS
         OCTAL      777633710004,.MRELS
14205    OCTAL      14206710000,.44SM3
***EOF
```

$LOAD (GECOS MODULE SECTION)

Hard core modules may be entered at Startup by inclusion of the card deck in the GECOS module section. These modules take precedence over those contained on the system program files.

HCM modules must be floatable programs which have binary decks with relocatable card format. However, any relocation indicated by entries on the cards is ignored by the loader. External program references are regarded to be error conditions.

Sample GECOS Module Section

```
$LOAD

$      OBJECT

       module deck

$      DKEND
$      OBJECT

       module deck

$      DKEND
***EOF
```

SAMPLE INPUT

The control cards below illustrate typical Startup input decks. The first group of cards might be used to control system generation on a machine which contains no GECOS III System. The system edit section is included to initialize the file system catalogs and establish the system files. The second group of cards could be used to control system generation for following usage of the established catalogs and files. In this case, no file edit section is needed.

In both the sample input decks shown, the hardware configuration used is that shown below:

1 Processor
1 Memory Controller (64k memory)
1 IOC
1 Drum on Pub 0
1 Tape controller with 8 handlers crossbarred on Pubs 1 and 2
1 Disc controller with 2 disc units on Pub 4
1 Tape controller with 8 handlers crossbarred on Pubs 3 and 5
1 Card Reader on Pub 6
1 Card Reader on Pub 7
1 Typewriter on Pub 8
1 Printer on Pub 11
1 Printer on Pub 12
1 Card Punch on Pub 14

Bare Machine Input

1. BOOTSTRAP LOADER

2. STARTUP PROGRAM DECK

3. $CONFIG

```
$    SYID     TYP1
$    DATE     042667
$    MCT-0    64,PORT-0,IOC-0,PORT-1,PRO-0
```

```
$   IOC-0     PUB-0,DRUM*200,ST1
$   IOC-0     PUB-1,TAPE,UNITS-8
$   IOC-0     PUB-3,TAPE,UNITS-8,UNIT-3,MT3,
$   ETC       UNIT-4,MT4,UNIT-5,MT5
$   IOC-0     PUB-4,DISC*200,UNITS-2,UNIT-1,
$   ETC       DS1,UNIT-2,DS2
$   IOC-0     PUB-6,READER
$   IOC-0     PUB-7,READER
$   IOC-0     PUB-8,CONSOLE,TY1,TY2,TY3,TY4
$   IOC-0     PUB-11,PRINTER
$   IOC-0     PUB-12,PRINTER
$   IOC-0     PUB-14,PUNCH*100
$   XBAR      IOC-0,PUB-1,PUB-2
$   XBAR      IOC-0,PUB-3,PUB-5
***EOF
```

4. $EDIT

```
$   INIT      ST1,DS1,DS2
$   FILDEF    ST1,GECOS3/GECOS,20/Ø,SYS,MT3
$   FILDEF    ST1,GECOS3/SOFTWARE,25,SYS,*
$   FILDEF    ST1,GECOS3/SYSLIB,25,RDM,MT4
$   FILDEF    ST1,GECOS3/SYSINP,2/Ø
$   FILDEF    DS1,GECOS3/SYSOT1,400
$   FILDEF    DS2,GECOS3/SYSOT2,400
$   PERMCP    DS1,MT1
***EOF
```

5. $FILES

```
$   SYSTEM    GECOS3/GECOS,GECOS3SOFTWARE
$   SAVE      GECOS3/SYSINP
$   SYSOUT    GECOS3/SYSOT1
$   LIBRARY   RDM,GECOS3/SYSLIB
$   ACCOUNT   RMV,MT7
***EOF
```

6. ***EOF
   ***EOF

Initialize Machine Input

1. BOOTSTRAP LOADER

2. STARTUP PROGRAM DECK

3. $CONFIG

```
$   DATE      042767
$   MCT-0     64,PORT-0,IOC-0,PORT-1,PRO-0
$   IOC-0     PUB-0,DRUM*200,ST1
$   IOC-0     PUB-1,TAPE,UNITS-8
$   IOC-0     PUB-3,TAPE,UNITS-8,UNIT-3,MT3,
$   ETC       UNIT-4,MT4,UNIT-5,MT5
$   IOC-0     PUB-4,DISC*200,UNITS-2,UNIT-1,
$   ETC       DS1,UNIT-2,DS2
$   IOC-0     PUB-6,READER
$   IOC-0     PUB-7,READER
$   IOC-0     PUB-8,CONSOLE,TY1,TY2,TY3,TY4
$   IOC-0     PUB-11,PRINTER
$   IOC-0     PUB-12,PRINTER
$   IOC-0     PUB-14,PUNCH*100
$   XBAR      1OC-0,PUB-1,PUB-2
$   XBAR      IOC-0,PUB-3,PUB-5
***EOF
```

4. $FILES

```
$   SYSTEM   GECOS3/GECOS,GECOS3/SOFTWARE
$   SAVE     GECOS3/SYSINP
$   SYSOUT   GECOS3/SYSOT1
$   LIBRARY  RDM,GECOS3/SYSLIB
$   ACCOUNT  RMV,MT7
***EOF
```

5. ***EOF

```
***EOF
```

3. STARTUP PROCESSING

The Startup package is, in itself, a miniature operating system. It provides for the initialization of all file system devices; edits various files into the file system, so that a copy of the system and its associated files may be called into core; it is a system loader, generating the tables of module and file locations needed by the system; and finally, it constructs the resident Hard Core Monitor and turns control over to GECOS III.

GENERAL FLOW OF STARTUP

The general flow of Startup is as follows:

1. Manual initialization and bootstrap procedures on T & O panels.
2. The 11-card bootstrap and loader:
   - locates the card reader
   - clears memory
   - loads the Startup program
3. The system description card processor:
   - reads the system description and control cards
   - builds system description tables
   - sets up I/O parameters for Startup
   - does any editing that is requested
4. The module loading process:
   - loads and initializes permanent modules from system storage
   - builds the module entry table and table of modules by program number
   - makes additional initialization entries as requested by the modules
5. The final phase of Startup:
   - moves the module entry table and table of modules by program number to the communication region
   - moves interrupt vectors, fault vectors, IOC mailboxes, etc., to their proper locations
   - transfers control to the Dispatcher module

GECOS-III CORE STORAGE LAYOUT

Figure 13 shows the core layout for the GECOS-III system at the end of each Startup segment which alters the system layout.

The size of the area set aside for real-time IOC mailbox storage is an assembly parameter and is equal to 512 *.NRRIO where .NRRIO is equal to the number of real-time IOC's to be provided for.

The size of the area set aside for the IOC mailbox storage is derived from the hardware configuration control cards and is equal to 256 * C(.CRNIC) where C(.CRNIC) is equal to the contents of the parameter indicating the number of configured IOC's.

The size of the area set aside for processor fault vector storage is also derived from the hardware configuration control cards and is equal to 32 * C(.CRNPC) where C(.CRNPC) is equal to the contents of the parameter indicating the number of configured processors.

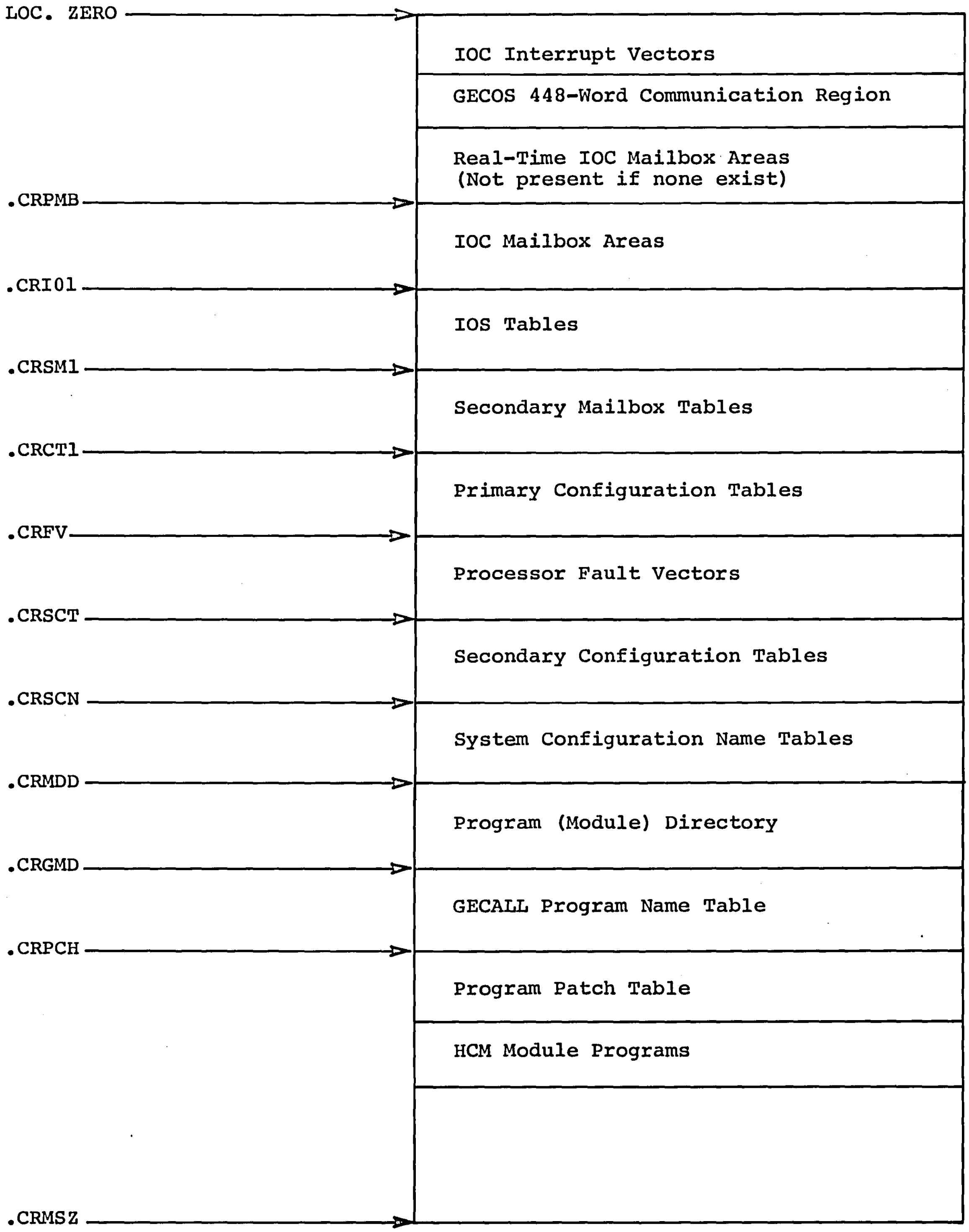


Figure 13. Core Layout

COMMUNICATION REGION SYMBOLS INITIALIZED BY STARTUP

SYMBOL	STARTUP SECTION INVOLVED
.CR9SA	$CONFIG
.CRACF	$EDIT
.CRBTS	Environmental Control
.CRCIC	$CONFIG
.CRCMC	$CONFIG
.CRCT1	$CONFIG
.CRCT3	$CONFIG
.CRCT4	$CONFIG
.CRDAT	$CONFIG (or operator reply)
.CRDIT	$FILES
.CRFV	Environmental Control
.CRGMD	$FILES
.CRI01	$CONFIG
.CRI04	$CONFIG
.CRI0C	$CONFIG
.CRLST	$FILES
.CRMCM	$CONFIG
.CRMDD	$CONFIG, $FILES, $LOAD
.CRMSZ	Environmental Control, $CONFIG
.CRNIC	$CONFIG
.CRNPC	$CONFIG
.CRPCH	$FILES, $PATCH
.CRPMB	$CONFIG
.CRQST	$FILES
.CRSCN	$CONFIG
.CRSCT	$CONFIG
.CRSYT	$FILES
.CRTCT	$CONFIG

STARTUP ABORTS

There are four general types of Startup aborts, as follows:

1. Startup comes to a stop after typing a terminal error message on the console.

   Refer to the GE-625/635 Typewriter Messages reference manual, CPB-1477, for a detailed description of the reason for the abort.

2. Startup comes to a stop with no external error indications.

   If this occurs, either:

   a. Force a post-mortem dump by depressing the EXECUTE button on the processor, or

   b. Examine location IC (170050) from the memory controller. This cell contains the instruction counter and indicators (IC & I) of where the fatal error occurred.

IC is	ERROR WAS
140050	Fault Vector switches set too low
140530	First CONFIG card was ***EOF
146040	End-of-file response from drum
157342	GECALL program name table full
163445	Illegal module number
164236	Illegal module number
164515	Dispatcher not loaded
166476	Console status bad (output)
166570	Console status bad (input)
167017	Needed device not in table

3. Startup automatically initiates a post-mortem dump with no external error conditions.

   Examine location IC when the dump has completed. This cell contains the IC & I of where the fatal error occurred.

IC is	ERROR WAS
164435	Number of processors configured is zero or greater than four
164451	Number of Input/Output Controllers configured is zero or greater than four

4. Startup automatically initiates a snapshot dump of the file system information area, immediately followed by a post-mortem dump.

   Index register 7 will contain the IC value describing the fatal file system error.

11-CARD BOOTSTRAP

Boot Card 1 is loaded into the IOC terminate interrupt address by a manual boot or into the terminate interrupt address -1 by a system boot. The routine is executed by the interrupt.

After execution of a system boot,

Location 0 contains the IOC MCT port number
X0 contains the IOC primary mailbox address
X2 contains the card reader pub number multiplied by 4

Boot Card 2 is loaded into the IOC terminate interrupt address and executed by the interrupt. For a manual boot, column 72 must contain the IOC MCT port number in binary in rows 8 and 9.

After execution of a manual or system boot,

X4 contains the IOC terminate interrupt address +2

After a system boot, location 0 and X0 and X2 remain unchanged.

QU contains the sum of the contents of X0 and X2

Boot Card 3 is loaded into the IOC terminate interrupt address and executed by the interrupt. This routine sets up a card read routine. The contents of location 0 and the registers remain unchanged.

Boot Card 4 is a continuation of the testing portion of the read routine set up by card 3.

Boot Card 5 contains the remainder of the read routine and the routine initialization. It sets up a read routine at location 4096.

Boot Cards 6-11 make up a loader which loads absolute binary cards and mask correction cards.

ENVIRONMENT CONTROL ROUTINE

STBEG begins Startup processing and establishes control of the hardware environment in which Startup is to operate.

INPUT DATA

LOC 0 - IOC primary mailbox address and
IOC memory controller port number
LOC 1 - IOC terminate interrupt vector entry address +2
LOC 2 - IOC channel number multiplied by 4

ENTRY

Entry is made to this routine at the end of boot card processing.

EXIT

Control is transferred to the Configuration Control Routine via TRA GETCF.

OUTPUT DATA

.CRFV - Processor fault vector origin
.CRMSZ - Physical core store size
.CRBTS - Address of the dump call instruction

METHOD

1. All I/O interrupts are disabled to prevent unexpected interrupts during Startup processing.

2. The card reader definition stored in core by the bootstrap loader is saved and stored in the Startup card read subroutine (READ).

3. The processor fault vector is located by setting all possible fault vector entries and executing a fault. It is set to contain the addresses of entries to the Startup fault routine.

4. Core storage outside the Startup program is cleared and the size of addressable core is found and stored in .CRMSZ.

5. Upon completion, control is sent to the Configuration Control Routine (GETCF).

CONFIGURATION CONTROL ROUTINE

GETCF processes the $CONFIG cards and constructs internal GECOS configuration and device control tables.

INPUT DATA

$CONFIG control cards
.CRFV Processor fault vector origin
.CRMSZ Physical memory size
.CRBTS Dump routine linkage

ENTRY

Control is transferred from STBEG via TRA GETCF.

EXIT

At the end of $CONFIG section processing, control is transferred to the $EDIT section via TZE GETED from the IEINP subroutine.

OUTPUT DATA

MCTMS - Memory size table
.CRMCM - Memory controller masks
.CRCIC - IOC connect addresses
.CRNIC - Number of IOS's
.CRSCT - Secondary configuration tables
.CRNPQ - Number of processors
.CRCMC - Processor connect addresses
.CRCT1 - Primary configuration tables
.CRCT3 - Primary configuration tables
.CRCT4 - Primary configuration tables
.CRI01 - IOS control
.CRI04 - IOS control
.CR9SA - 9SA control processor
.CRDAT - Date
.CRTCT - System trace control table
.CRMSZ - Memory size
.CRSCN - System name table
.CRIOC - IOC type indicator
.CRSID - System identification

METHOD

1. The next control card is read from the configuration section. If the card is a defined hardware configuration control card, control is sent to the subroutine which processes the card and constructs specified system parameters. If the card is not a defined hardware configuration card, the card image is typed, along with an error message, and the read is repeated.

2. The card processing routine scans the control card and accomplishes the specified initialization. Erroneous cards cause definitive error messages on the console typewriter. Control returns to step (1) above at the end of processing for each card.

3. Hardware configuration processing is terminated when the ***EOF card at the end of the hardware configuration section is reached.

$CONFIG CARD CLASSIFICATION

GETCD obtains the next $CONFIG control card, reads and classifies it, and initiates card processing.

INPUT DATA

$CONFIG control cards
CDTYT - CONFIG card type table

ENTRY

GETCD is entered at the end of processing for each $CONFIG card, except ***EOF, via TRA GETCD.

EXIT

Control is transferred to the correct card processing subroutine via

```
LDX2        CDTYT,DU
RPT         CDTTL/2,2,TZE
CMPA        0,2
TZE         -1,2*
```

When the ***EOF card is encountered, control is transferred via TRA IEINP.

METHOD

1. The next control card is read from the Startup input deck.
2. The card type field is scanned and tested to determine whether the card is a defined hardware configuration control card. Undefined control cards cause an error message and the card image to be typed on the console.
3. Control is transferred to the subroutine which processes the type of control card found.

System Id. Card	-	SYID
Date Card	-	IBDAT
Trace Card	-	IBTRC
MCT Card	-	IBPMC
IOC Card	-	IBPIO
XBAR Card	-	IBPXB
9SA Card	-	IBP9S
***EOF Card	-	IEINP

OTHER ROUTINES USED

DIS: Stop
GETCC: Next card control
SCBCD: BCD field scan
STCDC: Card image DCW control
TYPE: Typewriter output

SYSTEM IDENTIFICATION

SYID tests the $ SYID card for errors and stores the system identification in .CRSID.

INPUT DATA

$ SYID card

ENTRY

SYID is entered from GETCD.

EXIT

If no errors are found, control is returned via TRA GETCD.
If an error exists, control is transferred via TNZ CDUER or TZE FLERR.

OUTPUT DATA

.CRSID - System identification

METHOD

The identification field of the card is scanned and the value placed in .CRSID.

In the event of an error, duplicate cards (nonfatal) or punctuation (fatal), control is transferred to an error processing routine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

CDUER: Duplicate cards
FLERR: Fatal error
SCBCD: BCD field scan

DATE

IBDAT tests the $ DATE card (if any) for errors and stores the date in .CRDAT.

INPUT DATA

$ DATE card

ENTRY

IBDAT is entered from GETCD.

EXIT

If no errors are found, control is returned via TRA GETCD. If an error exists, control is transferred via TNZ CDUER or TZE FLERR.

OUTPUT DATA

.CRDAT - Date

METHOD

The date field of the card is scanned, and the value is placed in the system date word.

In the event of an error, duplicate cards (nonfatal) or punctuation (fatal), control is transferred to an error routine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

CDUER: Duplicate cards
FLERR: Fatal error
SCBCD: BCD field scan

TRACE

IBTRC tests the $ TRACE card for errors and stores the values in .CRTCT.

INPUT DATA

$ TRACE card

ENTRY

IBTRC is entered from GETCD.

EXIT

If no errors are found, control is returned via TNZ GETCD. If an error exists, control is transferred via TNZ CDUER, TZE FLERR, or TRA FLERR.

OUTPUT DATA

.CRTCT - Trace control table

METHOD

The trace control fields are scanned and the values entered in the system trace control table.

In the event of an error, duplicate cards (nonfatal), punctuation or too many fields (fatal), control is transferred to an error routine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

CDUER: Duplicate cards
FLERR: Fatal error
SCOCT: Numeric Field Scan

MCT

IBPMC tests the MCT cards for errors, enters the values in the memory controller mask and memory size tables, and establishes the origin for the secondary configuration tables.

INPUT DATA

$ MCT cards

ENTRY

IBPMC is entered from GETCD.

EXIT

If no errors are found, control is returned via TRA GETCD. If an error exists, control is transferred via TNZ CDUER, TNZ IOCTE, or TZE, TRC, or TNZ FLERR.

OUTPUT DATA

MCTMS - Startup core size table
.CRMCM - Memory controller mask
.CRCIC - IOC connect addresses
.CRNIC - Number of IOC's
.CRSCT - Secondary SCT origin
.CRCMC - Processor connect addresses
.CRNPC - Number of processors

METHOD

1. The memory controller number field of the card is scanned. The value obtained is used as the index to access memory controller mask and core size tables.

2. The core size field of the card is scanned. The value obtained is converted to the number of 512-word blocks and placed in the core size table. The core size is also accumulated for use by the configuration section end subroutine.

3. Memory controller port identification fields and associated module fields of the card are scanned. Each configured port causes a bit to be entered in the memory controller access mask at the appropriate place. Each attached IOC also causes the IOC interrupt mask bits to be included in the memory controller mask.

4. IOC and processor connect addresses are obtained from the card which describes memory controller number zero. Values for the number of IOC's and the number of processors are also found from this card. These values are used to establish the origin for the secondary configuration tables.

In the event of an error, duplicate cards (nonfatal), punctuation, illegal card entry, or duplicated card entry (fatal), control is transferred to an error routine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

SCNUM: Numeric field scan
SCBCD: BCD field scan
CDUER: Duplicate cards
FLERR: Fatal error

IOC

IBPIO tests the $ IOC cards for errors, begins building the configuration tables, and builds the Startup device name table.

INPUT DATA

$ IOC cards

ENTRY

IBPIO is entered from GETCD.

EXIT

If no errors are found, control is returned via TRA or TNZ GETCD. If an error exists, control is transferred via TNZ CDUER, or TZE, TNZ, TNC, or TRC FLERR.

OUTPUT DATA

IBANT	-	Device name table
.CRCT1 .CRCT3 .CRCT4	-	IOC entries to primary SCT
.CRSCT	-	IOC entries to secondary SCT

METHOD

1. The IOC and channel number fields of the card are scanned. These values are used as the index to access the configuration tables in the IOC mailbox area. The IOC number field is also used to set the IOC type indicator.

2. The device type field is scanned and tested. Device associated parameters are set to control allowable number of units, unit number limits, and allowable number of modules. These parameters control interpretation of the validity of following entries on the card.

3. When the number of units is determined, the primary and secondary configuration tables are constructed.

4. Remaining unit description fields are processed. Unit names are entered in the Startup device name table.

5. Alterations are made in the configuration tables to reflect other unit description entries.

6. In the event of an error, channel already configured (nonfatal), mixed IOC types, punctuation, illegal card entry, or duplicated unit name (fatal), control is transferred to an error routine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

CDUER:	Duplicate cards
FLERR:	Fatal error
IOCTE:	Illegal IOC type
SCBCD:	BCD field scan
SCNUM:	Numeric field scan
BLDCN:	Build configuration tables

XBAR

IBPXB tests the $ XBAR cards for errors and builds a table of crossbarred channel addresses used by the IOS control tables.

INPUT DATA

.CRCT1, .CRCT3, .CRCT4 - Primary configuration tables
$ XBAR cards

ENTRY

IBPXB is entered from GETCD.

EXIT

If no errors are found, control is returned via TRA GETCD. If an error exists, control is transferred via TNZ CFERR, TZE XBCER, or TZE, TNZ, TNC, or TRC FLERR.

OUTPUT DATA

.CRI01 - IOS control tables
.CRCT1, .CRCT3, .CRCT4 - Primary configuration tables

METHOD

1. The channel identification fields of the card are scanned. The values obtained are used to construct an internal table of crossbarred channel addresses.

2. When all card fields have been processed, the internal channel address table is used to establish the IOS control tables for the crossbarred channels. The primary configuration tables for the non-primary crossbarred channels are copied from those for the primary channel.

3. In the event of an error, primary PUB undefined (nonfatal), punctuation, illegal card entry, PUB field duplicated, too many PUB fields, too few PUB fields, or secondary channel configured (fatal), control is transferred to an error routine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

CFERR: Duplicate configuration
FLERR: Fatal error
SCBCD: BCD field scan
SCNUM: Numeric field scan
XBCER: Undefined primary channel

9SA

IBP9S tests the $ 9SA card for errors and inserts the processor number in the IOS tables.

INPUT DATA

$ 9SA card

ENTRY

IBP9S is entered from GETCD.

EXIT

If no errors are found, control is returned via TRA GETCD. If an error exists, control is transferred via TNZ CDUER, or TZE, TNZ, or TRC FLERR.

OUTPUT DATA

.CR9SA - Control processor number

METHOD

1. The processor identification field of the card is scanned. The value obtained is used to set the internal 9SA control processor number.

2. In the event of an error, card duplication (nonfatal), punctuation or illegal card entry (fatal), control is transferred to an error subroutine which produces a console message and returns control directly to GETCD.

OTHER ROUTINES USED

CDUER: Duplicate cards
FLERR: Fatal error
SCBCD: BCD field scan
SCNUM: Numeric field scan

CONFIGURATION CARD DUPLICATION

CDUER causes an error message to be typed when duplicate $CONFIG cards are encountered.

ENTRY

CDUER is entered from SYID, IBPMC, IBP9S, IBDAT, IBTRC, and IBPIO.

EXIT

Control is returned via TRA GETCD.

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

TYPE: Typewriter output

DUPLICATE CONFIGURATION

CFERR causes an error message to be typed when an $ XBAR card specifies a channel which has already been configured.

ENTRY

CFERR is entered from IBPXB.

EXIT

Control is returned via TRA GETCD.

OUTPUT DATA

ERRFL - Fatal error flag

METHOD

The fatal error flag is set.

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

TYPE: Typewriter output

CONFIGURATION CARD ERROR

FLERR causes an error message to be typed when a $CONFIG card is encountered which contains a fatal error; for example, values in the variable field too high or too low, or punctuation error.

ENTRY

FLERR is entered from SYID, IBPMC, IBPXB, IBP9S, IBDAT, IBTRC, and IBPIO.

EXIT

Control is returned via TRA GETCD.

OUTPUT DATA

ERRFL - Fatal error flag

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

GTCOL: Card column identification
TYPE: Typewriter output

IOC TYPE ERROR

IOCTE causes an error message to be typed when an $ MCT card specifies the wrong model IOC.

ENTRY

IOCTE is entered from IBPMC.

EXIT

Control is returned via TRA GETCD.

OUTPUT DATA

ERRFL - Fatal error flag

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

TYPE: Typewriter output

UNDEFINED PRIMARY CHANNEL

XBCER causes an error message to be typed when an $ XBAR card specifies a primary channel which has not been defined.

ENTRY

XBCER is entered from IBPXB.

EXIT

Control is returned via TRA GETCD.

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

TYPE: Typewriter output

CONFIGURATION SECTION END

IEINP completes the hardware configuration processing after all CONFIG control cards have been entered.

INPUT DATA

.CRMSZ	-	Core size
.CRCT1	-	Primary configuration table
.CRNIC	-	Number of IOC's
ERRFL	-	Fatal error flag
IBANT	-	System name table
STMSZ	-	Addressable core size

ENTRY

IEINP is entered from GETCD via TRA IEINP.

EXIT

Control is transferred to the $EDIT section routine via TZE GETED.

OUTPUT DATA

.CRMSZ	-	Core size
.CRSCN	-	System name table
.CRMDD	-	Module directory
.CRSCT	-	Secondary configuration table length
.CRI04 and .CRI01	-	IOS control
.CRIOC	-	IOC type indicator

METHOD

1. The size of configured core is compared with the physical addressable core and the smaller of the two values is placed in the core size parameter.

2. The secondary configuration table length is used to establish the system name table origin and the name table is moved from Startup to system storage.

3. The primary configuration entries for all IOC channels are tested. Channels which are not configured are eliminated from use by entry of the control bit in the IOS control table for the channel.

4. The system name table length is used to establish the origin for the program directory and the basic structure of the program directory is moved from startup to system storage.

5. If Model IOC-C configuration cards were found, the IOC type indicator is set to indicate this.

6. The fatal error flag is tested. If fatal configuration errors were encountered during processing, a message is typed and Startup is aborted.

OTHER ROUTINES USED

DIS: Stop
TYPE: Typewriter output

FILE EDIT CONTROL ROUTINE

GETED processes the $EDIT cards and performs the specified edit functions.

INPUT DATA

$EDIT control cards
System program tape files
Tape data files

ENTRY

GETED is entered from IEINP via TZE GETED.

EXIT

At the end of normal $EDIT section processing, control is transferred to the $FILES section via TRA NDEDT from ELNTP.

If an error occurs during edit processing, ECERR transfers control to DIS.

OUTPUT DATA

.CRDAT is set by console entry if no $ DATE card
System program random files
Random data files
File system space tables
File system catalogs

METHOD

1. The next control card is read from the input deck. If the card is not a file edit section descriptor, the file edit segment is bypassed.

2. The next control card is read from the file edit control section. If the card is a defined EDIT control card, control is sent to the routine which processes the card and accomplishes the edit functions. If the card is not a defined EDIT card, the contents of the card are typed, along with an error message and the read is repeated.

3. The processing routine scans the control card and executes the edit function. Erroneous cards cause definitive messages on the console typewriter. Control returns to step (2) above at the end of processing for each control card.

4. File edit processing is terminated when the ***EOF card marking the end of the EDIT section is encountered.

$EDIT CARD CLASSIFICATION

GETEC obtains the next control card from the $EDIT section, reads and classifies the card, and initiates card processing.

INPUT DATA

```
.CRDAT  -  Date
ECTYT   -  EDIT card type table
$EDIT control cards
```

ENTRY

GETEC is entered at the end of normal processing for each $EDIT card, except ***EOF, as shown below.

```
TNZ     GETEC (INIT)
TRA     GETEC (FILDEF and DUMP)
TZE     GETEC (PERMCP)
```

EXIT

Control is transferred to the correct card processing subroutine via

```
LDX2    ECTYT,DU
RPT     ECTYT/2,2,TZE
CMPA    0,2
TZE     -1,2*
```

When the ***EOF card is encountered, control is transferred via TRA ELNTP.

METHOD

1. The system date cell is tested and if the date has not been entered by the hardware configuration section, it is requested through the console.

2. The next control card is read from the Startup input deck. The card type field is tested to determine if the card is a defined file edit control card. Undefined control cards cause an error message and the card image to be typed on the console; processing halts.

3. Control is transferred to the routine which processes the type of control card identified.

4. Control is sent to ELNTP when the ***EOF card at the end of the file edit section is encountered.

GETEC
.MINIT

OTHER ROUTINES USED

TYPE: Typewriter output
KEYIN: Typewriter input
GETCC: Next card control
STCDC: Card image DCW control

INIT PROCESSING

IN10 determines the random access device and sets device-dependent data as specified on the $ INIT cards.

INPUT DATA

$ INIT cards

ENTRY

IN10 is entered from GETEC.

EXIT

If no errors are found, control is returned via TNZ GETEC. If an error exists, control is transferred via TRA ECERR.

OUTPUT DATA

X2 - Points to device SCT
X6 - Points to device name table
Available LINK and LLINK tables defined

METHOD

1. The first device name entry is scanned and the value obtained used to set the PAT for I/O and the PAT pointer.
2. Master catalogs and LINK and LLINK tables are defined and written on random access storage.
3. When more than one device name appears on the INIT card, steps 1 and 2 are repeated for each name entered.
4. If an error occurs, control is transferred to ECERR, a console message is typed, processing halts, and Startup is aborted.

OTHER ROUTINES USED

SCBCD: BCD field scan
STSCT: Set SCT pointer
STDVP: Set device pointer
ECERR: Edit error
RNDIO: Random I/O

SUPPORTING INFORMATION

The file system random storage is divided into two types of segments referred to as LLINKS and LINKS.

LLINKS

A LLINK is used to contain the file system space control tables and the file system catalogs. LLINKS 0-34 are dedicated to use for the space control tables and the subcatalogs which make up the master catalog. The following shows how the space is allocated:

LLINK NUMBER	USAGE
1	Available LLINK Table
2	Available LINK Table
3 - 34	Subcatalogs of Master Catalog

Other LLINKS are used as needed for the file system catalogs.

LINKS

A LINK is used to contain the files themselves. System program and library files are files of this type which are always recorded in a specific format. There are no dedicated LINKS.

FILDEF PROCESSING

FI10 creates the system catalog, reserves file space, and reads tape records onto the random access device as specified on the $ FILDEF cards.

INPUT DATA

$ FILDEF control cards
System tape files

ENTRY

FI10 is entered from GETEC.

EXIT

If no errors are found, control is returned via TRA GETEC. If an error exists, control is transferred via TZE, TNZ, or TRA ECERR.

OUTPUT DATA

X2 - Points to device SCT
X4 - Points to edit tape SCT
X6 - Points to device name table
File names in catalog

METHOD

1. The next control card is scanned, the SCT pointer stored in PAT, and the device pointer set.
2. File names are inserted into the catalog and tested for override flag. The file and its format are defined and then copied into the system.
3. Steps 1 and 2 are repeated for each FILDEF card.
4. In the event of an error, control is transferred to ECERR, a console message is typed, processing halts, and Startup is aborted.

OTHER ROUTINES USED

SCBCD:	BCD field scan
STSCT:	Set SCT pointer
STDVP:	Set device pointer
ECERR:	Edit error
SCNUM:	Numeric field scan
POSIT:	Tape position
RNDOM:	Disc or Drum I/O
TAPE:	Tape I/O
TYPE:	Typewriter Output
RNDIO:	Random I/O

PERMCP PROCESSING

PER10 copies files written in GECOS-II Permfile format and stores file names in the catalog.

INPUT DATA

$ PERMCP control cards
GECOS-II Permfiles

ENTRY

PER10 is entered from GETEC.

EXIT

If no errors are found, control is returned via TZE GETEC. If an error exists, control is transferred via TZE or TNZ ECERR, or, in the event of incorrect tape format, XED DIS.

OUTPUT DATA

X2 - Points to device SCT
X6 - Points to device name table
File names in catalog

METHOD

1. The next control card is scanned, the SCT pointer stored in PAT, and the device pointer set.
2. The card is checked for the presence of an option, the tape format is verified, the files are read into the system, and the file names placed in the catalog.
3. In the event of an error, control is transferred to ECERR, a console message is typed, processing halts, and Startup is aborted.

OTHER ROUTINES USED

SCBCD:	BCD field scan
STSCT:	Set SCT pointer
STDVP:	Set device pointer
ECERR:	Edit error
POSIT:	Tape position
SDUMP:	Snapshot dump
TYPE:	Typewriter output
RNDOM:	Disc or drum I/O
DIS:	Stop
TAPE:	Tape I/O

FILE DUMP PROCESSING

FLSDP dumps file system catalogs and random files on the printer. It is used as a utility function in system checkout.

INPUT DATA

$ DUMP control card

ENTRY

FLSDP is entered from GETEC.

EXIT

If no errors are found, control is returned via TRA GETEC. If an error exists, control is transferred via TNC or TZE ECERR.

OUTPUT DATA

Printer dump

METHOD

1. The device name field is scanned and the value obtained is used to set the dump heading, PAT table for I/O, and the device parameter table pointer.

2. The file segment type field is scanned to determine the type of file segment to be accessed and the segment number where the dump is to begin.

3. The size field is scanned to determine the size of the file segment to be dumped.

4. Elements of the file segment (LLINKS or LINKS) are read into core from the file and dumped on the printer by the snapshot dump (SDUMP).

5. The steps above are repeated for each set of control card entries. Control is returned to GETEC when the end of the dump card is reached.

6. In the event of an error, control is transferred to ECERR, a console message is typed, processing halts, and Startup is aborted.

FLSDP
.MINIT

OTHER ROUTINES USED

CONVD:	Decimal conversion
ECERR:	Edit error
PRINT:	Printer output
RNDIO:	Random I/O
SCBCD:	BCD field scan
SCNUM:	Numeric field scan
SDUMP:	Snapshot dump
STDVP:	Set device pointer
STSCT:	Set SCT pointer

EDIT CARD ERROR

ECERR causes an error message to be typed when an $EDIT card contains an error.

ENTRY

ECERR is entered from IN10, FI10, PER10, and FLSDP.

EXIT

Control is transferred via XED DIS.

METHOD

An error message, along with the incorrect card image, if any, is typed.

Startup is aborted by transferring control to DIS.

OTHER ROUTINES USED

GTCOL: Card column identification
TYPE: Typewriter output
DIS: Stop

EDIT SECTION END

ELNTP rewinds the tapes, zeros the PAT pointer, and transfers control to NDEDT.

INPUT DATA

LNTAB - File system link control table
***EOF card

ENTRY

ELNTP is entered from GETEC.

EXIT

Control is transferred to the $FILES processing routine via TZE or TRA NDEDT.

OUTPUT DATA

File system space control tables to random storage

METHOD

The available LINK storage table status is tested to determine if the table for a particular device is currently in core. If so, the table is written onto random storage.

The input tape status is tested to find whether the last used input unit is positioned off the load point. If so, the tape is rewound.

OTHER ROUTINES USED

POSIT: Tape position

SYSTEM FILE CONTROL ROUTINE

NDEDT processes the $FILES control cards and builds the internal program directory and file control tables.

INPUT DATA

$FILES control cards
System program random files
File system catalogs
FCTYT - File card type table

ENTRY

If no EDIT cards are in the deck, control is transferred to NDEDT from GETED via TRA NDEDT. After existing EDIT cards are processed, control is transferred from ELNTP via TRA NDEDT.

EXIT

At the end of $FILES section processing, control is transferred to the PATCH section via TRA or TZE PRPCH from ENFCF. If there are no $FILES cards in the deck, control is transferred via TRA PRPCH from NDEDT.

OUTPUT DATA

.CRLST - System library file control
.CRQST - System input file control
.CRDIT - System program file control
.CRMDD - Program directory
.CRSTY - System output file control
.CRGMD - GECALL program name table
.CRACF - System accounting file control
GECOS system map

METHOD

1. The next control card is read from the input deck. If the card is not a $FILES card, the System File Control routine is bypassed.
2. The next control card is read. If the card is a $FILES card, control goes to the routine which processes the card and initializes the appropriate system parameters. If the card is not a defined software configuration card, the card is typed with an error message and the read operation is repeated.
3. The card processing routine scans the control card, searches the file system catalogs and files for the specified entries, and builds the internal software system control tables. Erroneous cards cause definitive error messages on the console typewriter. Control returns to step (2) above at the end of processing for each control card.
4. System File Control processing is terminated when the ***EOF card at the end of the $FILES section is encountered.

$FILES CARD CLASSIFICATION

GETFC obtains the next control card from the $FILES section, reads and classifies the card, and initiates card processing.

INPUT DATA

$FILES control cards
FCTYT - File card type table

ENTRY

GETFC is entered at the end of normal processing for each $FILES card except ***EOF, as shown below.

```
TRA  GETFC (LIBRARY, ACCOUNT, SAVE, SYSOUT)
TNZ  GETFC (SYSTEM)
```

EXIT

Control is transferred to the correct card processing subroutine via

```
LDX2  FCTYT, DU
RPT   FCTTL/2,2,TZE
CMPA  0,2
TZE   -1,2*
```

When the ***EOF card is encountered, control is transferred via TRA ENFCF.

OUTPUT DATA

SCNTL - Card variable field scan tally

METHOD

1. The next control card is read from the Startup input deck.
2. The card type field is tested for a defined software configuration control card. Undefined control cards cause an error message and the card image to be typed on the console and the read to be repeated.
3. Control is transferred to the routine which processes the type of control card found.
4. Control is sent to the End of File Control subroutine when the ***EOF card marking the end of the software configuration section is reached.

OTHER ROUTINES USED

GETCC: Next card control
STCDC: Card image
TYPE: Typewriter output

SAVE FILE CARD PROCESSING

SYICD enters the name of the file to be used for dump control and restart procedures, and as a pushdown file during system initialization.

INPUT DATA

$ SAVE card

ENTRY

SYICD is entered from GETFC.

EXIT

If no errors are found, control is returned via TRA GETFC. If an error exists, control is transferred via TNZ FCDUP, TZE FCERR, or TRA FLUND.

OUTPUT DATA

.CRQST - System input file control

METHOD

1. The catalog search routine is used to find a file system catalog which describes the file named on the $ SAVE card.
2. The SCT and LINK number assigned to the file is entered into .CRQST.
3. A 1-LINK file is established on the slowest device. This file will be used for storing the sneak-on portion of dump.
4. The SCT and LINK number of the dump file is placed in the initialization linkage and made available to .MDUMP during its initialization.

OTHER ROUTINES USED

FCDUP: Duplicate cards
FCERR: File card error
FLUND: File undefined
SCBCD: BCD field scan

SYSTEM FILE CARD PROCESSING

SYSCD enters the system program files which make up the GECOS and software systems.

INPUT DATA

$ SYSTEM control cards
System program random files
File system catalogs

.CRMDD	-	Program directory
GNMTB	-	GECALL program name table
STNMB	-	GECOS module name table
SCNTL	-	Card variable field scan tally

ENTRY

SYSCD is entered from GETFC.

EXIT

If no errors are found, control is returned via TNZ GETFC. If an error exists, control is transferred via TNZ FCDUP, TZE FCERR, or TRA FLUND.

OUTPUT DATA

.CRDIT	-	System program file control
.CRMDD	-	Program directory
SFILE	-	System file name table
GNMTB	-	GECALL program name table

METHOD

1. The catalog search routine is used to find a file system catalog which describes the file named by the next entry on the card.

2. The file definition is obtained from the catalog and placed in the system program file control table with the storage device SCT pointer.

3. The file name is moved from the catalog to the Startup system file name table.

4. The catalog of elements blocks (FILESYS-2) are processed from the file found by the search of step (1) above. Each element name is tested to determine if the element is a GECOS module or a system program to be included in the GECALL table.

5. If an element is a GECOS module which has not yet been defined, the element description is placed in the program directory with the system program file control table index.

6. If an element is not a GECOS module it is assumed to be a system program eligible for use through GECALL. If the program has not yet been defined, the element description and the system program file control table index are used to construct an additional entry for the program directory. A corresponding entry is made in the Startup GECALL program name table.

7. Steps (1) through (6) above are repeated for up to eight system program files.

8. The following conditions cause a file card error routine to be used to produce a console error message. These error routines return control directly to the file card classification routine.

ERROR CAUSE	SEVERITY
Card Duplication	Non-fatal
Punctuation	Fatal
Too Many Card Entries	Fatal

9. When a persistent checksum error is encountered on random file access, an error message is written on the console and Startup is aborted.

OTHER ROUTINES USED

FCDUP: Duplicate card
FCERR: File card error
FLUND: Undefined file
SCBCD: BCD field scan
CONVD: Decimal conversion
PRINT: Printer output
RNDIO: Random I/O
TYPE: Typewriter Output
CATSR: Catalog Search

SYSOUT FILE CARD PROCESSING

SYOCD enters the names of up to four files to be used for collecting system output.

INPUT DATA

File system catalogs
SCNTL - Card variable field scan tally
$ SYSOUT card

ENTRY

SYOCD is entered from GETFC.

EXIT

If no errors are found, control is returned via TRA GETFC. If an error exists, control is transferred via TNZ FCDUP, TZE FCERR, or TRA FLUND.

OUTPUT DATA

.CRSYT - System output file control table

METHOD

1. The catalog search routine is used to find a file system catalog which describes the file named by the next entry on the card.
2. The file definition is obtained from the catalog and placed in the system output file control table with the storage device SCT pointer. The size of the smallest file is saved as the system output file size.
3. Steps (1) and (2) above are repeated for up to four system output files.
4. When all file names have been processed, the total amount of storage assigned to system output files is placed in the system output file control table. This value is the product of the smallest file size and the number of files described on the control card.
5. The following conditions cause a file card error routine to produce a console error message. These error routines return control directly to the file card classification routine.

ERROR CAUSE	SEVERITY
Card Duplication	Non-fatal
File Storage Devices not Same Type	Fatal
Too Many Card Entries	Fatal
Punctuation	Fatal

OTHER ROUTINES USED

```
CATSR:  Catalog search
FCERR:  File card error
FCDUP:  File card error
```

LIBRARY CARD PROCESSING

LIBCD enters the names of the files to be used for the system library.

INPUT DATA

$ LIBRARY Card
SCNTL - Card variable field scan tally
DVCDT - System storage device table

ENTRY

LIBCD is entered from GETFC.

EXIT

If no errors are found, control is returned via TRA GETFC. If an error exists, control is transferred via TNZ FCDUP, TRA or TZE FCERR, or TRA FLUND.

OUTPUT DATA

.CRLST - System library file control table

METHOD

The following procedure is used:

1. The catalog search routine is used to find a file system entry for the file name given on the card.

2. The file storage device type is obtained from the file system catalog and tested to determine if the file resides on system storage. If this is the case, the file definition is obtained from the catalog and placed in the library file control word along with the device SCT pointer.

3. If the library file does not reside on system storage, the removable device name is obtained from the catalog and converted into a device SCT pointer. This SCT pointer is placed in the library file control word along with the device name.

4. The following conditions cause a file card error routine to be executed to produce a console error message. These error routines return control directly to the file card classification routine.

ERROR CAUSE	SEVERITY
Card Duplication	Non-fatal
Punctuation	Fatal

LIBCD
.MINIT

OTHER ROUTINES USED

CATSR: Catalog search
FCDUP: Duplicate cards
FCERR: File card error
FLUND: Undefined file
SCBCD: BCD field scan
STSCT: Set SCT pointer

ACCOUNT CARD PROCESSING

SYACD enters the name of the file to be used to store accounting information.

INPUT DATA

$ ACCOUNT cards
SCNTL - Card variable field scan tally
DVCDT - System storage device table

ENTRY

SYACD is entered from GETFC.

EXIT

If no errors are found, control is returned via TRA GETFC. If an error exists, control is transferred via TNZ FCDUP, TRA or TZE FCERR, or TRA FLUND.

OUTPUT DATA

.CRACF - System accounting file control table

METHOD

The following procedure is used:

1. The catalog search routine is used to find a file system entry for the file named on the card.
2. The file storage device type from the file system catalog is tested to determine if the file resides on system storage. If this is the case, the file definition from the catalog is placed in the accounting file control word along with the device SCT pointer.
3. If the accounting file does not reside on system storage, the removable device name from the catalog is placed in the accounting file control word. The device SCT pointer is also found and placed in the file control word with the device name.
4. The following conditions cause a file card error routine to be used to produce a console error message. These error routines return control directly to the file card classification routine.

ERROR CAUSE	SEVERITY
Card Duplication	Non-fatal
Punctuation	Fatal

SYACD .MINIT

OTHER ROUTINES USED

```
CATSR:  Catalog search
FCDUP:  Duplicate cards
FCERR:  File card error
FLUND:  Undefined file
SCBCD:  BCD field scan
STSCT:  Set SCT pointer
```

DUPLICATE FILE CARD

FCDUP causes an error message to be typed when duplicate $FILES cards are encountered.

ENTRY

FLUND is entered from SYSCD, SYICD, SYOCD, LIBCD, and SYACD.

EXIT

Control is returned via TRA GETFC.

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

TYPE: Typewriter output

FILE CARD ERROR

FCERR causes an error message to be typed when a $FILES card contains an error.

ENTRY

FCERR is entered from SYSCD, SYICD, SYOCD, LIBCD, and SYACD.

EXIT

Control is returned via TRA GETFC.

OUTPUT DATA

ERRFL - Fatal error flag

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

GTCOL: Card column identification
TYPE: Typewriter output

FILE UNDEFINED

FLUND causes an error message to be typed when the specified file is undefined.

ENTRY

FLUND is entered from SYSCD, SYICD, SYOCD, LIBCD, and SYACD.

EXIT

Control is returned via TRA GETFC.

METHOD

A message describing the card error is typed on the console typewriter.

OUTPUT DATA

ERRFL - Fatal error flag

OTHER ROUTINES USED

TYPE: Typewriter output

FILES SECTION END

ENFCG moves the GECALL name table to the end of the module directory, defines patch table origin, and prints the system map.

INPUT DATA

.CRMDD - Program directory
***EOF card

ENTRY

ENFCF is entered from GETFC.

EXIT

Control is transferred to the $PATCH processing routine via TZE PRPCH.

OUTPUT DATA

.CRGMD - GECALL program name table
GECOS system map

METHOD

1. Uses tally word STMST, used previously to locate name table and module directory, to move GECALL name table to final location.
2. The setting of STMST after GECALL table is moved is used to define beginning of patch table.
3. The module directory and the GECALL name table are scanned. The data for the system map is extracted and printed.

OTHER ROUTINES USED

PRINT: Printer output
TYPE: Typewriter output
DIS: Stop

PATCH CONTROL ROUTINE

PRPCH processes the octal $PATCH control cards and builds the internal patch table.

INPUT DATA

$PATCH control cards
.CRGMD - GECALL program name table

ENTRY

PRPCH is entered from ENFCF via TZE PRPCH.

EXIT

At the end of $PATCH section processing, control is transferred to the $LOAD section via TRA XCARD from ENPCF.

OUTPUT DATA

.CRPCH - Program patch table
.CRMDD - Program directory

METHOD

1. The next control card is read from the input deck. If the card is not a program patch descriptor, the program patch segment is bypassed.

2. The next decimal card is read from the program patch section. If the card is an octal patch card, control is sent to the routine which processes the patch card and sets up the patch table.

3. The patch card processing routine scans the patch card and makes entries in the internal patch table. Erroneous cards cause definitive error messages on the console typewriter. Control returns to step (2) above at the end of processing for each card.

4. Program patch processing is terminated when the end of file card at the end of the program patch section is reached.

PATCH CARD CLASSIFICATION

GETPC obtains the next control card from the PATCH section, reads and classifies the card, and initiates card processing.

INPUT DATA

PATCH cards
.CRPCH - Program patch table

ENTRY

GETPC is entered at the end of processing of each PATCH card via TRA GETPC.

EXIT

Control is transferred to the processing subroutine via TZE OPCHC.

METHOD

1. PATCH section cards are read using the READ subroutine.
2. Binary cards are ignored. BCD cards are tested for EOF.
3. If the card is an EOF, control is transferred to ENPCF.
4. If the card is an OCTAL card, control is transferred to OPCHC for processing.
5. If neither EOF nor OCTAL, a message is typed indicating the card format is illegal and the card is ignored.

OTHER ROUTINES USED

READ:	Read card
STCDC:	Card image DCW control
TYPE:	Typewriter output

OCTAL CARD PROCESSING

OPCHC enters the load address, values to be loaded, and the name of the program to be altered.

INPUT DATA

LOFDT - Location field tally
MNTBT - Name table search tally
PCHPT - Patch table pointer
NXPCA - Next patch address

ENTRY

OPCHC is entered from GETPC via TZE OPCHC.

EXIT

If no errors are found, control is returned via TRA GETPC. If an error exists, control is transferred via TRA or TZE PCHNU, or TZE PCERR.

OUTPUT DATA

.CRMDD - Program directory

METHOD

1. This subroutine determines whether the module to be patched exists in the program directory or the GECALL name table. If in neither, an error message is typed and the card is ignored.

2. The patches contained on the card are entered into the patch table and the entry in the program directory or GECALL name table corresponding to the module to be patched is flagged to indicate a patch exists for it in the patch table.

OTHER ROUTINES USED

PCERR: Patch card error
PCHNU: Patched program undefined

PATCH CARD ERROR

PCERR causes an error message to be typed when a $PATCH card contains an error.

ENTRY

PCERR is entered from OPCHC.

EXIT

Control is returned via TRA GETPC.

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

GTCOL: Card column identification
TYPE: Typewriter output

PATCHED PROGRAM UNDEFINED

PCHNU causes an error message to be typed when the specified program is not defined.

ENTRY

PCHNU is entered from OPCHC.

EXIT

Control is returned via TRA GETPC.

METHOD

A message describing the card error, along with the contents of the incorrect card, is typed on the console typewriter.

OTHER ROUTINES USED

TYPE: Typewriter output

END OF PATCH FILE

ENPCF completes PATCH section processing and transfers control to the card LOAD section.

INPUT DATA

***EOF card

ENTRY

ENPCF is entered from GETPC.

EXIT

Control is transferred to the $LOAD section via TRA XCARD.

OUTPUT DATA

.CRPCH - Program patch table

METHOD

1. This subroutine constructs a tally for accessing the patch table and stores it in .CRPCH.

2. The starting location of HCM is defined.

CARD LOADER ROUTINE

XCARD enters hard core modules which will take precedence over those contained on the system program files.

INPUT DATA

$LOAD control cards
.CRMDD - Program directory

ENTRY

XCARD is entered at the end of Patch processing via TRA XCARD from ENPCF or from PRPCH when there are no $PATCH cards.

EXIT

At the end of $LOAD section processing, control is transferred to the System File Loader via TZE XBGEX.

OUTPUT DATA

.CRMDD - Program directory
GECOS HCM memory entries
GECOS memory map

METHOD

1. The next card is read from the input deck. If the card is a binary card or a control card other than $OBJECT, the card is ignored and the read is repeated.

2. When a $OBJECT card has been encountered, the binary cards of the program deck are read, and the program is loaded into the next available HCM space with an address which is a multiple of eight.

3. The program entry and initialization entry are defined by the first two SYMDEF entries of the program.

4. The program entry is defined by a symbol of the form ..xxxx. The address of this entry point is placed in the program directory and used when control is transferred to the module during GECOS execution. If no program entry is found, the module is overlayed by the next module which is loaded.

5. The initialization entry is defined by a symbol of the form .Ixxxx. Control is relinquished by the loader at the address of this entry for module initialization when the $DKEND card is encountered.

6. Processing is repeated at step (1) above when the $DKEND card has been reached for each program deck.

7. Control is sent to the module file load control segment when the end of file card at the end of the GECOS module section is reached.

XCARD .MINIT

OTHER ROUTINES USED

READ:	Card read
TYPE:	Typewriter output
KEYIN:	Typewriter input
XTYPI:	Card type illegal
CONVO:	Octal conversion
PRINT:	Printer output
DIS:	Stop

SUPPORTING INFORMATION

GECOS HCM modules may be initialized for execution when loaded by Startup. Following initialization, the memory space used by the initialization routine may be returned to the loader for use by the next module. Also, the initialization routine may designate additional required modules to be loaded. These functions are accomplished through use of the .ENTRY macro and a module initialization program which are included in the symbolic module deck.

.ENTRY MACRO

Module initialization is indicated by entry of the value INIT as argument number six of the entry macro within the module. This argument causes the displacement from the module entry address (..xxxx) to the initialization address (.Ixxxx) to be entered in bits 18-29 of the module entry word (..xxxx). This displacement is used to locate the initialization entry when the module is loaded from a system file.

Initialization Entry Conditions

1. Calling Sequence

   TSX1 .Ixxxx

2. Input Data

   X0 - Address of card reader definition

WORD 1:	ZERO PMB address, MCT port	
where:	PMB address	- IOC primary mailbox address
	MCT port	- IOC memory controller port number
WORD 2:	ZERO terminate address + 2	
where:	terminate address	- IOC terminate interrupt vector entry address
WORD 3:	ZERO PUB*4	
where:	PUB	- IOC channel number

   X2 - Address of IOC interrupt vector image
   X3 - Address of processor fault vector image
   X4 - Address of real-time IOC mailbox image

3. Usage

   Appropriate GECOS modules are required to use the input data to establish initial conditions for the items listed. These values cannot be initialized in place because Startup processing causes alteration.

4. System Status

   GECOS system communication region parameters and configuration tables are established in memory at the appropriate locations when module loading and initialization are executed.

Initialization Exit Conditions

1. Return

   0,1 when initialization complete

2. Output Data

   QU reg - next available load address
   A-reg - Required module list tally
   Module list entry ZERO module number

   where: module number - Number of module which must also be loaded

3. Usage

   The module loader unconditionally uses the output data described above to control the loading of following modules. Therefore, all module initialization routines must take care to establish this data on return to the loader.

SYSTEM FILE LOADER

XBGEX loads the required hard core modules from the system program files.

INPUT DATA

System program random files
.CRMDD - Program directory

ENTRY

XBGEX is entered after $LOAD processing is complete via TZE XBGEX.

EXIT

When system files have been loaded, control is transferred to the GECOS entry via TNZ XBDTF.

OUTPUT DATA

.CRMDD - Program directory
GECOS HCM memory entries
GECOS memory map

METHOD

1. The entries of the internal program directory are tested until one is found which describes a required HCM module that has not been loaded. Undefined, required module entries cause error messages on the console typewriter.
2. The required module is loaded into the next available HCM space with an address which is a multiple of eight.
3. The program entry address is found from the element control block and placed in the program directory.
4. If an initialization displacement exists in the entry word (..xxxx), control is relinquished by the loader at the initialization entry.
5. Processing is repeated at step (1) above when loading and initialization is complete for each module.
6. Control is sent to the GECOS entry segment when all program directory entries have been processed.

OTHER ROUTINES USED

STDVP:	Set device pointer
CONVO:	Octal conversion
PRINT:	Printer output
RNDIO:	Random I/O
DIS:	Stop
TYPE:	Typewriter output
CHKSM:	Checksum

SUPPORTING INFORMATION

See XCARD.

XBDTF
. MINIT

GECOS ENTRY

XBDTF provides for orderly transition from the Startup program to the GECOS Dispatcher to begin processing.

INPUT DATA

.CRNPC	-	Number of processors
.CRFV	-	Processor fault vector origin
.CRNIC	-	Number of IOC's
.CRPMB	-	IOC primary mailbox origin
.NRRIO	-	Number of real-time IOC's
.CRMCM	-	Memory controller masks
.CRMSZ	-	Memory size
.CRMDD	-	Program directory
.MDISP	-	Dispatcher module number
.CRCMC	-	Processor connect address table
STIVI	-	IOC interrupt vector image
STRII	-	Real-time IOC mailbox image
STFVI	-	Processor fault vector image
MCTMS	-	Memory controller memory size table

ENTRY

XBDTF is entered from XBGEX after system files have been loaded.

EXIT

Control of all processors is sent through the fault routine by generation of a connect fault in each processor. This transfers control from Startup to GECOS.

OUTPUT DATA

.CRFV	-	Processor fault vectors
.CRPMB	-	IOC mailboxes

METHOD

1. The actual IOC interrupt vector is set from the Startup interrupt vector image.

2. If provision is made in system assembly for real-time IOC's, the real-time mailboxes are set from the Startup real-time mailbox image.

3. The processor fault vectors are set from the Startup fault vector image.

4. The first and third blocks of the IOC mailboxes are initialized. These areas contain the primary and secondary mailbox entries for the IOC and the interrupt queues.

5. The interrupt and access masks are set in all configured memory controllers to allow interrupt and control signals on only the configured memory parts.

6. If the GECOS Dispatcher module has not been found and loaded during the HCM load processing, Startup is aborted by execution of the Stop routine.

7. If the GECOS Dispatcher is present, memory used by Startup is cleared and all processors are sent to GECOS through execution of connect faults.

CPB-1489

INTERNAL SUBROUTINES

In addition to the program segments constituting the main body of the Startup Program, there are a number of internal routines which provide frequently needed functions. These routines are located throughout Startup and are described on the following pages.

BUILD CONFIGURATION TABLES

BLDCN builds the configuration tables from the $ IOC cards.

INPUT DATA

IOC mailbox index

ENTRY

BLDCN is called from IBPIO via TSX1 BLDCN.

EXIT

If no errors are found, control is returned via TRA 0,1. If an error exists, control is transferred via TNZ CDUER.

OUTPUT DATA

.CRCT1,.CRCT3,.CRCT4 - IOC entries to primary SCT

METHOD

1. This subroutine constructs the 4-word table entries using the description on the card just read. If a PUB is erroneously defined more than once, a message is typed and the card is ignored.

2. The primary SCT entries are located as a function of the configuration; that is, the IOC primary mailbox index and the PUB being described immediately follow the IOC mailbox area. The secondary SCT's are located consecutively, as needed, immediately after the processor fault vector areas. This location is defined as:

   Processor number*32+IOC number*256+PMB origin.

OTHER ROUTINES USED

CDUER: Duplicate cards

CHECKSUM

CHKSM verifies the checksums of modules loaded from the system files. The checksum macro expansion provides for program blocks exceeding 256 words.

INPUT DATA

X3	-	Address of data block origin
Q-register	-	Data block length
CKSUM	-	Expected data block checksum
CKERC	-	Checksum error try count

ENTRY

CHKSM is called from XBGEX via TSX1 CHKSM.

EXIT

Control is returned via TZE 1,1 when checksum is good. In the event of an error, control is transferred via TNC 0,1 to retry.

METHOD

1. The data block checksum is calculated through use of the standard checksum macro expansion. This value is compared with the expected checksum and control returns to the calling program if the checksums agree.

2. When the checksums do not agree, the error count is incremented and compared with the error retry limit. If the limit has not been reached, control returns to the calling program for retry. Otherwise, a message is typed and Startup is aborted through execution of the Stop routine.

OTHER ROUTINES USED

DIS: Stop
TYPE: Typewriter output

DECIMAL CONVERSION

CONVD converts binary values to decimal form for output.

INPUT DATA

A-register - Binary value to be converted

ENTRY

CONVD is called from FLSDP, SYSCD, and ENFCF via TSX1 CONVD.

EXIT

Control is returned via TRA 0,1.

OUTPUT DATA

Q-register - Decimal value (leading zero character replaced with blanks.)

METHOD

1. Each leading decimal digit is converted and tested for zero. If zero, a blank is inserted in its place.
2. Remaining decimal digits are converted to form a value in the range $-10^6 < n < 10^6$ and control is returned to the calling program.

OCTAL CONVERSION

CONVO converts binary values to octal form for output.

INPUT DATA

Q-register - Binary value to be converted

ENTRY

CONVO is called from XCARD, XBGEX, FAULT, and DUMP via TSX1 CONVO.

EXIT

Control is returned via TRA 0,1.

OUTPUT DATA

A-register - Octal representation of the most significant 18 bits of the binary value

Q-register - Octal representation of the least-significant 18 bits of the binary value, shifted to the most-significant position.

METHOD

1. The binary value is converted by repeated left shifts until six octal characters are formed.

2. Control is returned to the calling program.

DEFINE LINK TABLES

DFLTB defines the available LINK and LLINK tables.

INPUT DATA

Q-register - Maximum available LINKS/LLINKS on device being initialized
X3 Beginning location of LINK or LLINK table
A-register Availability of the first 36 LINKS or LLINKS

ENTRY

DFLTB is called from IN10 via TSX7 DFLTB.

EXIT

Control is returned via TRC 0,7.

METHOD

1. This subroutine constructs an image of the available LINK or LLINK table in memory, starting at the location pointed to by X3 at entry.

2. The A-register defines the availability of the first 36 LINKS/LLINKS and sets all other bits corresponding to legitimate LINKS/LLINKS to zero (available). Nonexistent LINKS/LLINKS for this device are set to one for the remainder of the table.

DISC TRANSMIT COMMANDS

DISC accomplishes all input/output operations on DSU200 devices. The device to be used is specified by the PAT indicated in the call.

ENTRY

DISC is called during EDIT processing via

```
TSX1      DISC
ZERO      PAT pointer
Command
DCW
ZERO      Seek control, Block number
```

where:

Command		- Read or write instruction to be issued
DCW		- Must contain an IOTD or a TDCW
Seek control	=	-5, Absolute block within LLINKS
		-4, Absolute block within LINKS
		-2, Relative block within LLINK description
		-1, Relative block within LINK description
		+n, Seek address
		(LINK and LLINK descriptions are contained within the PAT specified by the PAT pointer when control values -2 and -1 are used.)
Block number		- Block number for data

EXIT

Control is returned to 4,1 for end of file, to 5,1 for all other cases.

METHOD

1. A calling sequence is constructed for the IOC interface routine, the unit address is found from the PAT table entry, the seek address is calculated, and the seek command is issued.

2. The status of the seek operation is tested. If the status is Channel Ready, the I/O command is issued. Otherwise, control goes to the proper major status routine as in step (4) below.

3. When the I/O command is issued, the status is again tested. If the status is Channel Ready, control is returned to the calling program.

4. If the Seek or I/O operation status is not Channel Ready, the unit message is set to contain the disc unit address and control goes to the proper major status routine.

   Attention

   All Substatus: Operator attention message is typed, console response delay is executed, Seek and I/O operation are re-issued.

Data Alert

Invalid Control Character: Illegal Major Status processing is used.

All other Substatus: The Seek and I/O operations are reissued four times in an attempt to achieve successful operation. If this fails, Illegal Major Status processing is used.

End of File

Control is returned to the calling program at the end of file return point.

Illegal Status

Stop message is typed and Startup is aborted.

5. If an error is encountered in seek address computation, the unit message is set to contain the disc unit address, a stop message is typed, and Startup is aborted.

OTHER ROUTINES USED

STSUA:	Unit address control
ISTIO:	IOC interface
STUNM:	Unit message control
TYPE:	Typewriter output
KEYIN:	Typewriter input
DIS:	Stop

DRUM TRANSMIT COMMANDS

DRUM accomplishes all input/output operations on MDU200 devices. The device to be used is specified by the PAT indicated in the call.

ENTRY

DRUM is called during EDIT processing via

```
     TSX1        DRUM
     ZERO        PAT pointer
     Command
     DCW
     ZERO        Select control, Block number
```

where:

```
     Command          - Read or write instruction to be issued
     DCW              - Must contain an IOTD or a TDCW
     Select control = -5, Absolute block within LLINKS
                      -4, Absolute block within LINKS
                      -2, Relative block within LLINK description
                      -1, Relative block within LINK description
                      +n, Select address

                      (LINK and LLINK descriptions are contained within
                      the PAT specified by the PAT pointer when control
                      values -2 and -1 are used.)

     Block number     - Block number for data
```

EXIT

Control is returned to 4,1 for end of file, to 5,1 for all other cases.

METHOD

1. A calling sequence is constructed for the IOC interface routine, the unit address is found from the PAT table entry, the select address is calculated, and the Select command is issued.

2. The status of the Select operation is tested. If the status is Channel Ready, the I/O command is issued. Otherwise, control goes to the proper major status routine as in step (4) below.

3. When the I/O command is issued, the status is again tested. If the status is Channel Ready, control is returned to the calling program.

4. If the Select or I/O operation status is not channel ready, the unit message is set to contain the drum unit address and control goes to the proper major status routine.

   Attention

   All Substatus: Operator attention message is typed, console response delay is executed, Select and I/O operation are reissued.

Data Alert

Invalid Select Data: Illegal Major Status processing is used.

All other Substatus: The Select and I/O operations are re-issued four times in an attempt to achieve successful operation. If this fails, Illegal Major Status processing is used.

End of File

Control is returned to the calling program at the end-of-file return point.

Illegal Status

Stop message is typed and Startup is aborted.

5. If an error is encountered in select address computation, the unit message is set to contain the drum unit address, a stop message is typed, and Startup is aborted.

OTHER ROUTINES USED

STSUA:	Unit address control
ISTIO:	IOC interface
STUNM:	Unit message control
TYPE:	Typewriter output
KEYIN:	Typewriter input
DIS:	Stop

CORE STORAGE DUMP

DUMP produces a terminal dump of all or selected portions of the core store. SDUMP produces snapshot dumps during Startup processing.

INPUT DATA

.CRMDD - Module directory
A-register - Parameter list tally
  Parameter: VFD 18/a,1/s, 1/p, 1/d, 15/n

where:
- a - Beginning address
- s - Slew control
  - =0, Slew to top of page at beginning
  - ≠0, Suppress slew
- p - Panel dump control
  - =0, Dump panel at beginning
  - ≠0, Suppress panel dump
- d - Dump control
  - =0, Dump core
  - ≠0, Suppress core dump
- n - Word count
  - =0, All memory
  - ≠0, Number of words to dump

ENTRY

DUMP is called from STBEG and XBDTF via XED DUMP. SDUMP is called from PER10 and FLSDP via XED SDUMP.

EXIT

From DUMP, control does not return to the calling routine. The next program to be executed is bootstrapped from the card reader. From SDUMP, control is returned via RET IC.

METHOD

1. For a terminal dump, the IOC mailboxes are saved before alteration by dump processing.
2. The next dump control parameter is tested. A printer slew is issued if specified. The hardware registers are dumped if panel dump is specified. The indicated section of core is dumped if core dump is specified.
3. Step (2) above is repeated for each dump control parameter in the list.
4. For a snapshot dump, control is returned to the calling program. For a terminal dump, the program is delayed until a special interrupt occurs on the card reader used to load Startup. At that point, the next program to be executed is bootstrapped from the card reader.

OTHER ROUTINES USED

PRINT: Printer output
TYPE: Typewriter output

FAULT PROCESSING

FAULT controls faults which occur in the control processor during Startup.

INPUT DATA

FLTIC - IC and indicators at fault

ENTRY

FAULT is entered from STBEG via

```
     STC1        FLTIC
     TRA         FAULT +n
```

EXIT

Startup is aborted by transferring control to DIS.

METHOD

Entry to the fault routine is made through the fault vector when a fault occurs. The fault vector is initialized at the beginning of Startup processing by the environment control segment.

1. A code is set in the fault message to indicate the type of fault.
2. The IC value is converted and set in the fault message.
3. The fault message is typed on the console typewriter.
4. Control is sent to the Startup stop routine.

OTHER ROUTINES USED

CONVO: Octal conversion
DIS: Stop
TYPE: Typewriter output

DEVICE LOCATION

FNDVC finds a configured device using either the specified device name or device type.

INPUT DATA

```
IBANT  - Startup device name table
.CRCT1 - System configuration table
.CRSCT - Secondary system configuration table
.CRNIC - Number of IOC's
.CRCIC - IOC connect address table
```

ENTRY

FNDVC is called from TYPE, KEYIN, and PRINT via

```
     TSX7  FNDVC
     VFD   18/device code,18/..
     VFD   #18/device name,18/..
```

EXIT

Control is returned to -1,7 when device address is set.

OUTPUT DATA

```
-1,7  -  LDX1 IOC number plus MCT port number multiplied by 8, DU
0,7   -  IOC primary mailbox address
1,7   -  PUB number multiplied by 4
```

METHOD

1. The Startup device name table is searched for an entry containing the same name as that specified in the call. If found, step (4) below is executed next.

2. If the device name specified in the call does not exist, if the name is assigned to a device of a type other than that specified in the call, or if the named device is not assignable, a search is made of the system configuration tables for an available device of the specified type. If found, step (4) below is executed next.

3. If an available device is not found by name table and configuration table search, a standard device address is selected from the device location table.

4. The device address is converted to the form needed for the IOC interface and stored in the calling sequence as indicated above. Control is then returned to the calling program.

WORKING STORAGE AREA INITIALIZATION

FSINIT initializes the file system working storage area and initiates I/O to read the master catalog.

INPUT DATA

X2 - Communication register
X3 - Address of working storage

ENTRY

FSINIT is called from FSIN01 and FSIN03 via TSX2 FSINIT,$.

EXIT

Control is returned via TRA 0,2

METHOD

Using the information provided by the linkage to the file system routines, FSINIT constructs a skeleton of the working storage area and fills in applicable data. The format and contents of the working storage area are detailed in the System Tables (CPB-1488).

FILE/CATALOG CREATE

FSIN01 and FSIN02 creates a file and the catalogs necessary to access it.

ENTRY

FSIN01 and FSIN02 are called from FI10 via

```
TSX4    FSIN01
NOP
ZERO    0,ARGLIST
ZERO    3,BUFFER
```

where:

ARGLIST contains a set of pointers to a return area, user name, catalog/file description, permissions, and option area.

BUFFER points to an area used as working storage.

EXIT

Control is returned to the proper routine via TRA 0,AU.

OUTPUT DATA

Return area	Contains completion codes (create successful or error code)
User name	Originator identification
Catalog/file description	Name and password
Permissions	Defines access permission
Option	Defines file type, device type, name, and file size

METHOD

Upon return from the create function, the contents of word 1 of the return area indicates whether the create has been successful, or if not, the error code associated with the reason for failure.

OTHER ROUTINES USED

FSDVDP: Device dependent initialization

FSIN03
.MINIT

FILE ACCESS AND RETRIEVE

FSIN03 performs a search for a necessary file.

ENTRY

FSIN03 is called from FI10 via

```
TSX4    FSIN03
NOP
ZERO    0,ARGLIST
ZERO    4,BUFFER
```

where:

ARGLIST contains a set of pointers to a return area, user name, catalog/file description, permission, and option area.

BUFFER points to an area used as working storage.

EXIT

Control is returned to the proper routine via TRA 0,AU.

OUTPUT DATA

Return area	Contains completion codes (create successful or error code)
User name	Originator identification
Catalog/file description	Name and password
Permissions	PAT pointer to access data
Option	Defines file type, device type, name, and file size

METHOD

Upon return from the search function, the contents of word 1 of the return area indicates whether the search has been successful, or if not, the error code associated with the reason for failure.

OTHER ROUTINES USED

FSDVDP: Device dependent initialization

NEXT CARD CONTROL

GETCC obtains the next control card from the Startup input deck.

INPUT DATA

GOTNC - Next card control word, as follows:

=0, next card not yet read
≠0, next card read and stored according to this control word, ZERO A,B

where: A is beginning address of card image
B is length of card image.

ENTRY

GETCC is called from any process cards subroutine in Startup via TSX1 GETCC.

EXIT

Control is returned via TZE ** for end of file and via TRA 1,1 for all other cases.

OUTPUT DATA

QU	-	Card image address
QL	-	Word count
TYFDT	-	Type field tally
VAFDT	-	Variable field tally
GOTNC	-	If zero, next card not yet read

METHOD

1. The next card control word is tested to determine whether the next control card or end of file card has been read by the scan routines. If so, step (2) below is executed. Otherwise, cards are read until an end of file card is encountered or a decimal card with a dollar sign character in column one is found.

2. Card field scan tally words are set and control is returned to the calling program.

OTHER ROUTINES USED

READ: Card read

GET CARD COLUMN NUMBER

GTCOL identifies the current position of the card scan.

INPUT DATA

SCNTL - Card image scan tally

ENTRY

GTCOL is called from FLERR, ECERR, FCERR, and PCERR via TSX1 GTCOL.

EXIT

Control is returned via TRA 0,1.

OUTPUT DATA

Q-register - Two-character BCD column number, left-justified with trailing zeros.

METHOD

1. The current address of the scan is found from the card image scan tally. The origin of the card image is found from the card image DCW. These values and the character position of the scan tally determine the current card column for the scan.

2. The card column number is converted from binary to BCD and positioned in the output register.

IOC INTERFACE

ISTIO is used for all access to the input/output hardware in Startup.

INPUT DATA

X0 - IOC primary mailbox address
X1 - IOC number plus MCT port number multiplied by 8
X2 - PUB number multiplied by 4 plus unit number multiplied by 64

ENTRY

ISTIO is entered from READ, DISC, DRUM, POSIT, TAPE, TYPE, KEYIN, and PRINT via

```
TSX7        ISTIO
Device command
DCW
```

where:

Device command is the peripheral instruction to be issued.

DCW must contain an IOTD or a TDCW.

EXIT

Control is returned via TRA 2,7.

OUTPUT DATA

X0 - Status return word pointer

METHOD

This routine processes requests for I/O action from beginning to end before control is returned to the calling program. The first and third blocks of the IOC mailbox area are affected by this routine and the IOC. The GECOS configuration and I/O control tables located in the second and fourth blocks of the mailbox area are not affected.

1. The IOC and software interrupt counters are cleared.
2. The primary and secondary mailboxes are initialized to accomplish the desired I/O operation.
3. The initiate, terminate, and special interrupt vectors are set.
4. Initiate, terminate, and special interrupts are enabled and a connect is issued to the IOC.
5. The processor is delayed until an initiate or terminate interrupt occurs from the proper channel.
6. A pointer is set to the interrupt queue entry containing the status of the attempted operation, all interrupts are disabled, and control is returned to the calling program.

SPECIAL INTERRUPT PROCESSING

ISTSI delays processing until a special interrupt occurs on the specified IOC and channel.

INPUT DATA

X0 - IOC primary mailbox address
X1 - IOC number plus MCT port number multiplied by 8
X2 - PUB number multiplied by 4 plus unit number multiplied by 64

ENTRY

ISTSI is called from READ, POSIT, TAPE, and PRINT via

```
TSX7        ISTSI
```

EXIT

Control is returned to 0,7 when the special interrupt occurs on the specified channel.

OUTPUT DATA

X0 - Status return word pointer

METHOD

The first block of the IOC mailbox area is affected by this routine and the IOC. The GECOS configuration and I/O control tables located in the second and fourth blocks of the mailbox area are not affected.

1. The IOC and software interrupt counters are cleared.
2. The special, initiate, and terminate interrupt vectors are set.
3. Initiate, terminate, and special interrupts are enabled.
4. The processor is delayed until a special interrupt occurs from the proper channel.
5. A pointer is set to the interrupt queue entry from the special interrupt, all interrupts are disabled, and control is returned to the calling program.

TYPEWRITER INPUT

KEYIN accomplishes input operations on the console typewriter. The console used is determined by the device location routine.

ENTRY

KEYIN is called from GETED, XCARD, DISC, and DRUM via

```
TSX1     KEYIN
DCW
```

where:

DCW - Must contain an IOTD or a TDCW

EXIT

Control is returned via TRA 1,1.

METHOD

1. A calling sequence is constructed for the IOC interface routine.
2. On the first call to typewriter input, the console to be used is found by the device location routine. This is done by a search of the system configuration for an assignable and non-dedicated console. Tests are made for a device with the name TY1. If this fails, a search is made for any available console. If this also fails, the standard console is selected from the device location table. The normal standard entry is PUB 8 on IOC 0.
3. The typewriter read command is issued.
4. The status of the read command is tested. If the status is Channel Ready, control is returned to the calling program.
5. If the read operation status is not Channel Ready, the unit message is set to contain the console address and control goes to the proper major status routine.

   Attention

   A write alert command is issued to the console to turn on the operator alarm. The input operation is then reissued.

   Data Alert

   Transfer Timing, Operator Error, Message Length Alert: A message is typed directing the operator to reenter the message and the input operation is reissued.

   Operator Distracted: The input operation is reissued.

   All Other Substatus: Illegal Major Status processing is used.

   Illegal Status

   Startup is aborted.

OTHER ROUTINES USED

FNDVC:	Device location
ISTIO:	IOC interface
STUNM:	Unit message control
DIS:	Stop
TYPE:	Typewriter output

TAPE POSITIONING

POSIT accomplishes tape positioning operations not requiring data transmission. The device to be used is specified by the PAT indicated in the call.

ENTRY

POSIT is entered from FI10, PER10, ELNTP, and TAPE via

```
TSX1      POSIT
ZERO      PAT pointer
Command
```

where:

Command - Nondata transmission instruction to be issued.

EXIT

Control is returned to 2,1 for end of file, to 3,1 for all other cases.

METHOD

1. A calling sequence is constructed for the IOC interface routine, the unit address is found from the PAT table entry, and the command is issued.
2. The status of the operation is tested. If the status is Channel Ready, control is returned to the calling program.
3. If the operation status is not Channel Ready, the unit message is set to contain the tape unit address and control goes to the proper major status routine.

Attention

Blank Tape on Write: Illegal Major Status processing is used.

All other Substatus: Operator attention message is typed, special interrupt delay is executed, operation is reissued.

Instruction Rejected

Load Point: Channel Ready major status processing is used.

All other Substatus: Illegal Major Status processing is used.

End of File

Control is returned to the calling program at the end of file return point.

Illegal Status

Stop message is typed and Startup is aborted.

POSIT
.MINIT

OTHER ROUTINES USED

STSHA:	Unit address control
ISTIO:	IOC interface
STUNM:	Unit message control
TYPE:	Typewriter output
ISTSI:	Special interrupt delay
DIS:	Stop

PRINTER

PRINT accomplishes printer output operations. The printer used is determined by the device location routine.

ENTRY

PRINT is called from FLSDP, SYSCD, ENFCF, XCARD, XBGEX, and DUMP via

```
TSX1    PRINT
DCW
```

where: DCW - Must contain an IOTD or a TDCW.

EXIT

Control is returned via TRA 1,1.

METHOD

1. A calling sequence is constructed for the IOC interface routine.
2. On the first call to print, the device to be used is found by the device location routine. This is done by a search of the system configuration for an assignable and nondedicated printer. Tests are made for a printer with the name PR1. If this fails, a search is made for any available printer. If this also fails, the standard printer is selected from the device location table. The normal standard entry is PUB 10 on IOC 0.
3. The print command is issued.
4. The status of the print operation is tested. If the status is Channel Ready, control is returned to the calling program.
5. If the print operation status is not Channel Ready, the unit message is set to contain the printer address and control goes to the proper major status routine.

   Attention

   VFU Tape Alert: Channel Ready major status processing is used.

   All other Substatus: Operator attention message is typed, special interrupt delay is executed, print operation is reissued.

   Data Alert

   Transfer Timing Error: Print operation is reissued.

   All other Substatus: Channel Ready major status processing is used.

Instruction Rejected

Previous Slew or Top of Page: Print operation is reissued.

All other Substatus: Illegal Major Status processing is used.

Illegal Status

Stop message is typed and Startup is aborted.

OTHER ROUTINES USED

FNDVC:	Device location
ISTIO:	IOC interface
STUNM:	Unit message control
TYPE:	Typewriter output
ISTSI:	Special interrupt delay
DIS:	Stop

READ CARDS

READ reads the next card from the Startup input deck.

ENTRY

READ is called from STBEG, GETCC, GETPC, and XCARD via

```
TSX1     READ
ZERO     buffer address
```

where:

buffer address = address of an area at least 24 words in length.

EXIT

Control is returned to 1,1 for a decimal card; 2,1 for a binary card.

OUTPUT DATA

Q-register - Card control word, in the form:

```
ZERO     B,L
```

where:

B - Beginning address of card image
L - Length of card image

METHOD

1. A calling sequence is constructed for the IOC interface routine and a mixed mode card read command is issued.
2. The status of the read operation is tested. If the status is Channel Ready, the card control word is set and control is returned to the calling program.
3. If the read status is not Channel Ready, the unit message is set to contain the card reader address and control goes to the proper major status routine.

   Attention

   Initiate Interrupt

   Card Jam or Sneak Feed: Stop message is typed and Startup is aborted.

   Manual Halt or Hopper Alert: Operator attention message is typed, special interrupt delay is executed, read operation is reissued.

   Terminate Interrupt

   Read Alert: Backspace message is typed, special interrupt delay is executed, read operation is reissued.

Feed Alert: Operator attention message typed, special interrupt delay is executed, read operation is reissued.

Last Batch, Manual Halt, Hopper Alert: Channel ready processing is used.

Card Jam: Stop message is typed and Startup is aborted.

Data Alert

Transfer Timing Error: Backspace message is typed, special interrupt delay is executed, read operation is reissued.

Illegal BCD Character: Illegal character backspace message is typed, special interrupt delay is executed, read operation is reissued.

Illegal Status

Stop message is typed and Startup is aborted.

OTHER ROUTINES USED

DIS:	Stop
ISTIO:	IOC interface
ISTSI:	Special interrupt delay
STUNM:	Unit message control
TYPE:	Typewriter output

RANDOM INPUT/OUTPUT

RNDIO sets up and executes calling sequences to the random I/O routines. The calling sequences may be modified by end-of-file procedures so are initialized before each use.

INPUT DATA

Startup random device tables

ENTRY

RNDIO is called from IN10, FI10, GIVER, DUMP, SYSCD, FS10, FSAARG, XCARD, and XBGEX via

```
TSX7     RNDIO
VFD      18/PAT pointer, 16/0, 1/read-write, 1/EOF control
DCW
ZERO     seek control, block number
```

where:

```
read-write     =0, Write instruction
               ≠0, Read instruction
EOF control    =0, No end-of-file recovery
               ≠0, End-of-file recovery
DCW - Data control word; must contain an IOTD or a TDCW.
seek control   = -5, Absolute block within LLINKS
                 -4, Absolute block within LINKS
                 -2, Relative block within LLINK description
                 -1, Relative block within LINK description
                 +n, Seek address

                 (LINK and LLINK description are contained within the
                 PAT specified by the PAT pointer when control values
                 -2 and -1 are used.)

block number   - Block number for data
```

EXIT

Control is returned via TRA 3,7 when I/O operation is complete.

METHOD

1. The PAT pointer, read or write command, seek address control parameter, and end of file recovery are not set up in the I/O calling sequence as specified.
2. The data control word is set up in the I/O calling sequence and modified to read the next hardware block if an IOTD with zero word count is given.
3. The I/O operation is accomplished by transfer of control to the proper I/O routine, and control is returned to the calling program when the operation is complete.

DISC OR DRUM I/O

RNDOM sets up and executes calling sequences to the random I/O routines. The calling sequences may be modified by end-of-file procedures so are initialized before each use.

INPUT DATA

Startup random device tables

ENTRY

RNDOM is called from FIl0, PERl0, and SCTRD via

```
          TSX7      RNDOM
          VFD       18/PAT pointer, 18/0, 1/read-write, 1/EOF control,
                    n/n
          DCW
          ZERO      seek control, block number
```

where:

```
     read-write     =0, Write instruction
                    ≠0, Read instruction
     EOF control    =0, No end-of-file recovery
                    ≠0, End-of-file recovery
     n/n            = Number of words
     DCW - Data control word; must contain an IOTD or a TDCW
     seek control   = -5, Absolute block within LLINKS
                      -4, Absolute block within LINKS
                      -2, Relative block within LLINK description
                      -1, Relative block within LINK description
                      +n, Seek address

                      (LINK and LLINK description are contained within the
                      PAT specified by the PAT pointer when control values
                      -2 and -1 are used.)

     block number   - Block number for data
```

EXIT

Control is returned via TRA 2,7 when I/O operation is complete.

METHOD

1. The PAT pointer, read or write command, seek address control parameter, and end of file recovery are not set up in the I/O calling sequence as specified.

2. The data control word is set up in the I/O calling sequence and modified to read the next hardware block if an IOTD with zero word count is given.

3. The I/O operation is accomplished by transfer of control to the proper I/O routine, and control is returned to the calling program when the operation is complete.

BCD FIELD SCAN

SCBCD scans control card images for BCD fields and break conditions.

INPUT DATA

SCNTL - Data image scan tally

ENTRY

SCBCD is called from all card processing subroutines via

```
TSX1        SCBCD
```

EXIT

Control is returned at field end or break condition via TRA **.

OUTPUT DATA

AQ-register - Next BCD field, left-justified with trailing blanks.

Break character switches are set as follows:

If Switch	$\neq 0$	Field ends on
CECSW		Comma
CEBSW		Blank
CESSW		Slash
CEDSW		Dash
CETSW		Tally runout

METHOD

1. The first character of the field is tested. For blank, the next control card is read and tested for ETC. If the next card is an ETC card, the scan is continued at the beginning of the variable field of the ETC card.

2. Field break indicators are all turned off and characters of the next field are accumulated until a break character is found or until 12 characters are accumulated.

3. The proper field break indicator is set and control is returned to the calling program.

NUMERIC FIELD SCAN

SCNUM and SCOCT (octal conversion) scan control card images for numeric fields and break conditions.

INPUT DATA

- SCNTL - Data image scan tally

ENTRY

SCOCT is called from IBTRC and OPCHC via TSX1 SCOCT.

SCNUM is called from all other card processing routines via TSX1 SCNUM.

EXIT

Control is returned at field end or break condition via TRA **.

OUTPUT DATA

Q-register - Numeric value converted to binary

Break character switches are set as follows:

If Switch	$\neq 0$	Field ends on
CECSW		Comma
CEBSW		Blank
CESSW		Slash
CEDSW		Dash
CETSW		Tally runout

METHOD

1. The conversion radix is set for octal or decimal conversion.
2. The first character of the field is tested. For blank, the next control card is read and tested for ETC. If the next card is an ETC card, the scan is continued at the beginning of the variable field of the ETC card.
3. Field break indicators are all turned off and digits of the next field are converted until a break character is found or until 12 octal digits or 10 decimal digits are processed.
4. The proper field break indicator is set and control is returned to the calling program.

OTHER ROUTINES USED

GETCC: Next card control

SET CARD IMAGE DCW

STCDC sets up the DCW used to output a card image.

INPUT DATA

Q-register - Card control word in the form ZERO B,L

where:

B - Beginning address of card image
L - Length of card image

ENTRY

STCDC is called from GETCD, GETEC, GETFC, and GETPC via TSX1 STCDC.

EXIT

Control is returned via TRA 0,1.

OUTPUT DATA

CDIDC - Card image DCW

METHOD

1. The incomplete word at the end of the card is filled with blank characters.
2. Trailing blanks are suppressed by adjusting the card control word count.
3. The adjusted word count and address from the card control word are used to build an IOTD to transmit the nonblank portion of the card.

SET DEVICE POINTER

STDVP performs the table search required to set up a device pointer when the configuration table entry is known.

INPUT DATA

A-register	-	Logical device name
X0	-	Configuration table pointer
.CRCT1	-	System configuration table
.CRSCT	-	Secondary system configuration table
DVCDT	-	Startup random device table

ENTRY

STDVP is called from IN10, FI10, PER10, FLSDP, FSDVDP, XCARD, and XBGEX via TSX1 STDVP.

EXIT

Control is returned via TRA 0,1 when the configuration table describes a random device handled by Startup.

If the table entry describes a device not handled by Startup, a message is typed and Startup is aborted.

OUTPUT DATA

X6 - Startup device table pointer

METHOD

1. The Startup random device table is searched for an entry containing the device code specified in the call.

2. If the device code is found in the random device table, the Startup device table pointer is set and control is returned to the calling program. Otherwise a message is typed on the console typewriter and Startup is aborted.

OTHER ROUTINES USED

DIS: Stop
TYPE: Typewriter output

GET SCT POINTER

STSCT performs the table search required to find a configuration table pointer when the device name is known.

INPUT DATA

.CRSCN - System name table
A-register - Logical device name

ENTRY

STSCT is entered from IN10, FI10, PER10, FLSDP, LIBCD, and SYACD via TSX1 STSCT.

EXIT

Control is returned via TRA 0,1 when the device name is found in the system name table. In the event the device name is undefined, an error message is typed and Startup is aborted.

OUTPUT DATA

X0 - Configuration table pointer

METHOD

1. The system name table is searched for an entry containing the logical device name specified in the call.

2. If device name is found in the system name table, the configuration table pointer is set and control is returned to the calling program. Otherwise, a message is typed on the console typewriter and Startup is aborted.

OTHER ROUTINES USED

DIS: Stop
TYPE: Typewriter output

SET UNIT ADDRESS

STSUA converts the unit address from configuration table format to the form used in the IOC interface calling sequence.

INPUT DATA

X2 - Unit address pointer
X3 - PAT table pointer

ENTRY

STSUA is called from DISC, DRUM, POSIT, and TAPE via TSX7 STSUA.

EXIT

Control is returned via TRA 0,7.

OUTPUT DATA

0,2	- IOC number plus MCT port number multiplied by 8
1,2	- IOC primary mailbox address
2,2	- PUB number multiplied by 4, plus unit number multiplied by 64.
.CRCIC	- IOC connect address table

METHOD

1. The unit, channel, and IOC numbers are derived from the configuration table entry pointed to by the PAT table.
2. The MCT port number is obtained from the IOC connect address table.
3. The values above are transformed to the desired form and inserted in the storage cells addressed by the unit address pointer.

SET UNIT MESSAGE

STUNM is used by all input/output subroutines to set up the unit address message. This unit address message forms a part of the operator attention and Startup stop messages issued by I/O subroutines.

INPUT DATA

X1 - IOC number and MCT port number multiplied by 8
X2 - PUB number multiplied by 4, plus unit number multiplied by 64

ENTRY

STUNM is called from READ, DISC, DRUM, POSIT, TAPE, KEYIN, and PRINT via

```
        TSX7        STUNM
```

EXIT

Control is returned to 0,7 when unit message is set.

METHOD

The unit address specified in the call is converted to printable form and placed in the unit message.

TAPE TRANSMIT COMMANDS

TAPE accomplishes all input/output operations on magnetic tape devices. The device to be used is specified by the PAT indicated in the call.

ENTRY

TAPE is called from FI10 and PER10 via

```
TSX1      TAPE
ZERO      PAT pointer
Command
DCW
```

where:

Command - Read or write instruction to be issued
DCW - Must contain an IOTD or a TDCW

EXIT

Control is returned to 3,1 for end of file, to 4,1 for all other cases.

OUTPUT DATA

A-register - End-of-file character when an end-of-file return is made

METHOD

1. A calling sequence is constructed for the IOC interface routine, the unit address is found from the PAT table entry, and the I/O command is issued.

2. The status of the I/O operation is tested. If the status is Channel Ready, control is returned to the calling program.

3. If the operation status is not Channel Ready, the unit message is set to contain the tape unit address and control is transferred to the proper major status routine.

   Attention

   Blank Tape on Write: Illegal Major Status processing is used.

   All Other Substatus: Operator attention message is typed, special interrupt delay is executed, operation is reissued.

   Data Alert

   Bit Detected on Erase: Channel Ready major status processing is used.

   End of Tape: Illegal Major Status processing is used.

Parity: Noise record test is made by data control word string and status return processing. If a noise record, operation is reissued. Otherwise, the tape is repositioned and the operation reissued four times in an attempt to achieve successful operation. If this fails, Illegal Major Status processing is used.

All other Substatus: The tape is repositioned and the operation is reissued four times in an attempt to achieve successful operation. If this fails, Illegal Major Status processing is used.

End of File

The end-of-file character is set for return and control is returned to the calling program.

Illegal Status

Stop message is typed and Startup is aborted.

OTHER ROUTINES USED

STSUA:	Unit address control
ISTIO:	IOC interface
STUNM:	Unit message control
TYPE:	Typewriter output
ISTSI:	Special interrupt delay
POSIT:	Tape position
DIS:	Stop

TYPEWRITER OUTPUT

TYPE accomplishes output operations on the console typewriter. The console used is determined by the device location subroutine.

ENTRY

TYPE is entered from all processing routines which output operator messages.

```
        TSX1      TYPE
```

EXIT

Control is returned via TRA 1,1.

METHOD

1. A calling sequence is constructed for the IOC interface routine.
2. On the first call to type, the console to be used is found by the device location routine. This is done by a search of the system configuration for an assignable and nondedicated console. Tests are made for a device with the name TY1. If this fails, a search is made for any available console. If this also fails, the standard console is selected from the device location table. The normal standard entry is PUB 8 on IOC 0.
3. The typewriter write command is issued.
4. The status of the write command is tested. If the status is Channel Ready, control is returned to the calling program.
5. If the write operating status is not Channel Ready, control goes to the proper major status routine.

Attention

A write alert command is issued to the console to turn on the operator alarm. The type write command is then reissued.

Data Alert

Incorrect format: Channel Ready major status processing is used.

Transfer Timing or Transmission Parity Error: The type write command is reissued.

All other Substatus: Illegal major status processing is used.

Illegal Status

Startup is aborted.

OTHER ROUTINES USED

FNDVC: Device location
ISTIO: IOC interface
DIS: Stop

INDEX

FIGURE (continued)

FILES

SYSTEM

CPB-1489

Please cut along this line

DOCUMENT REVIEW SHEET

TITLE: GE-625/635 GECOS-III Startup

CPB #: 1489

From:

Name: ______________________

Position: ______________________

Address: ______________________

____

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

COMMENTS: ______________________

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

STAPLE STAPLE

FOLD

FIRST CLASS
PERMIT, No. 4332
PHOENIX, ARIZONA

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

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

ATTENTION: Program Documentation B-90
Systems and Processors Operation

FOLD

INFORMATION SYSTEMS

GENERAL ELECTRIC

LITHO U.S.A.