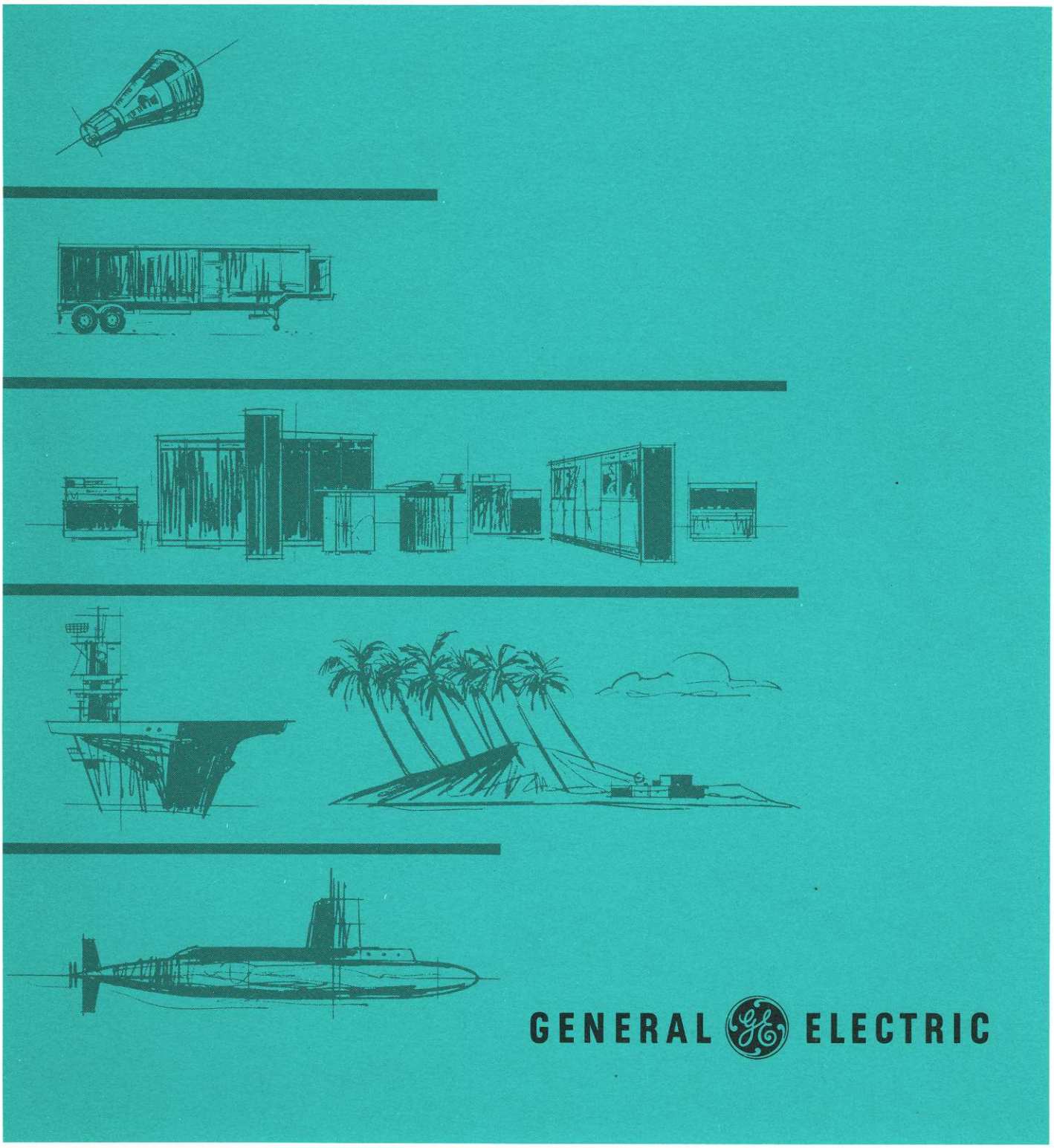
M-605 System Manual



GENERAL ELECTRIC

M-605 System Manual

GENERAL ELECTRIC

TABLE OF CONTENTS

I. INTRODUCTION I-1

A. General I-1
B. System Features I-1
1. Modularity I-1
2. Configurations I-2
3. Multi-Programming I-3
C. Programming Features I-5
D. Programming System I-5
E. Input/Output I-6

II. COMPUTER FEATURES II-1

A. Basic Modules II-1
1. Memory II-1
2. Processor II-2
3. Real-Time Input/Output Controller (RT-IOC) II-3
B. Configurations II-4
1. Basic II-4
2. Multi-Memory II-5
3. Multi-Processor II-5
4. Multi-Computer II-6
5. Control Processor Designation II-7
6. Reconfiguration II-7
C. Interrupts and Faults II-7
1. Program Interrupts II-7
2. Faults II-9

III. PROGRAMMING III-1

A. Word Formats III-1
1. Fixed-Point Data III-1
2. Floating-Point Data III-1
3. Alphanumeric Data III-1
B. Instruction Format III-1
C. Register Descriptions III-4
D. Address Modification III-4
1. Modification Types III-4
2. Character Operations III-7
E. Instruction Repertoire III-8
1. Data Movement III-9
2. Shifting III-9
3. Fixed-Point Arithmetic III-9
4. Boolean Operations III-9

TABLE OF CONTENTS (Cont.)

III (cont)
5. Comparison III-10
6. Floating-Point Arithmetic III-10
7. Transfer of Control III-11
8. Special Operations III-11

IV. PROGRAMMING SYSTEM IV-1

A. GECOS/605 . . . . . IV-1
B. GMAP IV-2
C. FORTRAN IV IV-2
D. GELOAD/605 . . . . . IV-3
E. Utility-Library Routines IV-3
F. Test and Diagnostic Programs IV-3
G. Support Software IV-4

V. PERIPHERAL EQUIPMENT SUBSYSTEMS V-1

A. Consoles V-1
B. Magnetic Tape Subsystem V-1
C. Card Reader Subsystem V-1
D. Card Punch Subsystem V-2
E. Line Printer Subsystem V-2
F. Paper Tape Subsystem V-2

APPENDIX A INSTRUCTION LISTING

APPENDIX B STANDARD GENERAL ELECTRIC CHARACTER SET

LIST OF ILLUSTRATIONS

Figure		Page
I-1	Multi-Processor System	I-2
I-2	Multi-Computer System	I-3
III-1	Fixed-Point Data Formats	III-2
III-2	Floating-Point Data Formats	III-2
III-3	Alphanumeric Data Formats	III-3
III-4	General Instruction Format	III-3
III-5	Indirect Word as Interpreted under I	III-5
III-6	Indirect Word as Interpreted under ID or DI	III-6
III-7	Indirect Word as Interpreted under SC	III-6
III-8	Indirect Word as Interpreted under CI	III-6
III-9	Indirect Word as Interpreted under AD	III-7
III-10	Indirect Word as Interpreted under IDC and DIC	III-7
III-11	Typical Character Operations	III-8

I. INTRODUCTION

A. GENERAL

The M-605 is a medium scale, military digital computing system designed for aerospace and defense applications. It is one member of the General Electric Compatibles/600 family which includes commercial and military, large- and medium-scale systems for business, scientific, and real-time applications, and fully compatible microminiaturized systems for airborne and spaceborne needs.

M-605 systems, with a full complement of software and peripheral devices, are designed for real time, ground-based, mobile, and shipborne applications with versions to meet the diverse needs of each environment.

The M-605 and its aerospace counterpart, the A-605, are fully compatible with larger scale 600-line systems. Their instruction repertoire and features are a subset of the larger systems. Programs may be compiled, assembled, and checked out on either the M-605 or at any available Compatibles/600 facility.

B. SYSTEM FEATURES

The M-605 is a modular computer system. Of particular importance in military real-time applications, multi-computer configurations can be formed to provide on-line failure detection and recovery for a high mission reliability.

1. Modularity

The computer is comprised of three primary, independent modules:

- Memory
- Processor
- Real-Time Input/Output Controller (RT-IOC)

These modules can be arranged in a variety of configurations using as many of the Memory modules as needed for the required storage, as many of the Processor modules as needed to provide the required computation capability, and as many RT-IOC modules as needed for the complement of peripheral and external devices included in the system.

Growth capability is assured; a system may be expanded by the addition of modules and connecting cables.

2. Configurations

The basic modules can be used to form either multi-processor or multi-computer configurations to meet the specific needs of each installation.

Multi-processor configurations, such as that shown in figure I-1, are used in installations in which the primary emphasis is on throughput. In such an application, all job scheduling, hardware allocation, input/output supervision, and housekeeping functions are performed by the software operating system, GECOS/605 (General Comprehensive Operating Supervisor), which is executed by only one of the several Processors that may be in the system. This Processor is considered the "Control" Processor with the other Processors subservient to it.

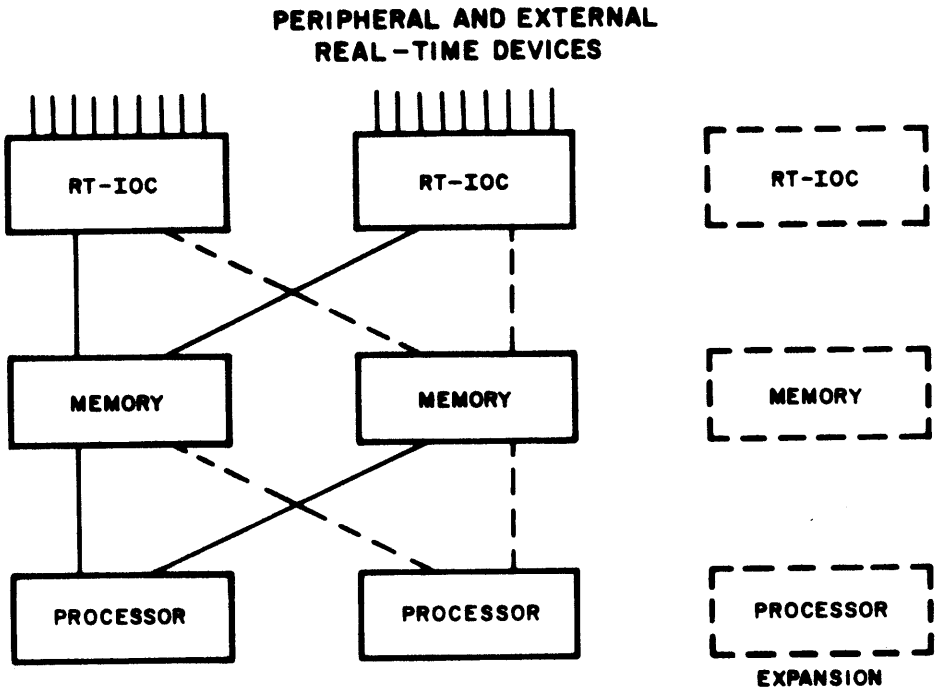


Figure I-1. Multi-Processor System

Multi-computer configurations, in which there are two or more Control Processors, are required where system reliability is of the utmost importance. A system of this type, shown in figure I-2, can be regarded as two or more separate computer systems each independent of the other but with facility for directly communicating between their respective Memories. The ability of either Processor to directly address data in either Memory or to execute programs in either Memory, coupled with system features for effective control, makes it possible for one system to serve as backup to the other, and provides for failure detection and recovery.

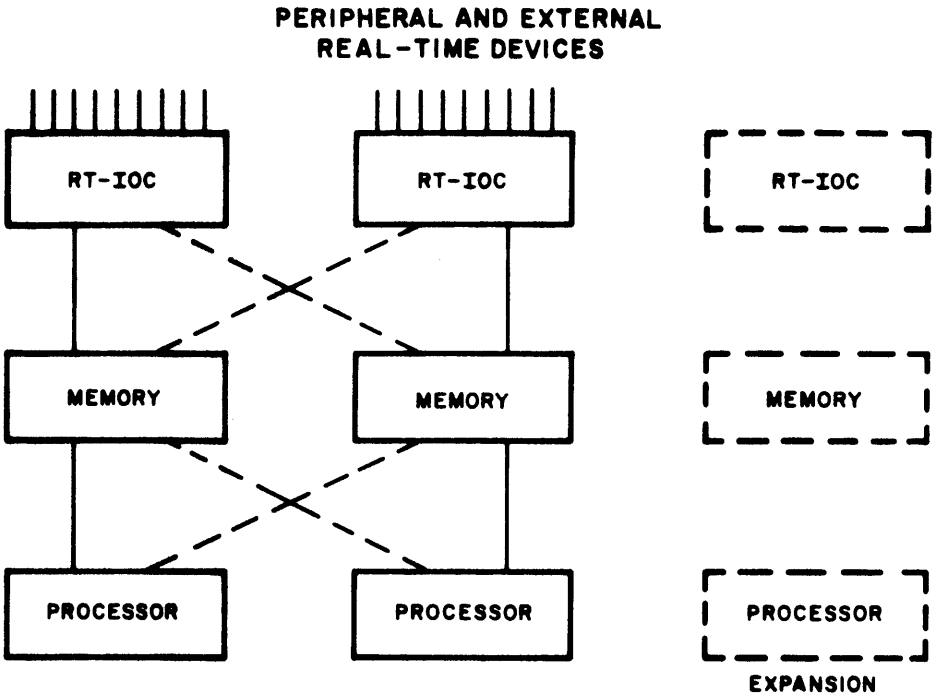


Figure I-2. Multi-Computer System

The configurations shown are only typical. Each Processor and RT-IOC can directly address up to four Memory modules or a total of 262,144 - 36 bit words of core storage. Up to eight devices, Processors or RT-IOC modules in any combination, may be connected to each Memory module.

3. Multi—Programming

Multi-programming is made possible in the 600-line computers through combined hardware and software techniques.

For real-time applications, multi-programming is mandatory to permit the computer to respond to demands made by the system in which the computer is used. The real-time system must perform concurrent computations within allowable time limits.

For non-real-time applications, multi-programming is necessary to maintain a high system efficiency, and thus high throughput, by optimizing equipment utilization. For example, on-line media conversion may be performed concurrently with on-line computations on a time-shared basis.

In either case, the program set may be considered as a number of independent programs to be executed in a time-shared manner. For the real-time mission, each independent program may correspond to a mission function and must complete its sequence of computations within a specified time after occurrence of the external event to which it must respond.

In the non-real-time case, each independent job program should time-share the computer until held up, usually because of the need for some input/output transaction to be completed, at which time the next most urgent program is executed.

a. PROGRAM INTERRUPTS

The ability of external devices to interrupt the program being executed by the Processor is essential to efficient multi-programming capabilities of the M-605.

The program interrupt facility acknowledges interrupts in priority order, interrupting the program only if the demand is of higher priority than the one that initiated the current program, as specified by the Mask Register. The priority can be changed at will by program. In the real-time environment, the interrupt may cause a transfer directly to the program that is to respond to the event without the intervention of an executive routine.

b. DUAL MODE OPERATION

In a multi-programming or time-sharing environment, a distinction must be made between control or executive programs and object programs. The executive program must be able to exercise control of the entire system and must be protected from inadvertent program errors.

The distinction is made in the M-605 by providing two Processor operating modes: Master and Slave. In the Master mode, no restrictions are imposed on the program. Programs executing in the Slave mode are inhibited from exercising control functions such as changing the interrupt priority, controlling I/O devices, setting the interval timer and other control registers. Further, memory protection is provided for the operating system by limiting the memory access by a slave program to certain boundaries specified by the operating system.

The dual operating mode provides positive protection for the system, insuring that each program in the system will not alter or affect the others.

c. DYNAMIC PROGRAM RELOCATION

Through a Base Address Register, object programs can be placed or moved anywhere in memory without the need for software relocation. Programs can be compacted in memory as individual programs are completed, allowing the system to use the released memory areas for new program allocation.

Each program is written and stored in memory with addresses relative to zero. Addresses are automatically relocated at execution time. Relocation is performed only in the Slave mode.

d. MEMORY PROTECTION

For object programs operating in the Slave mode, all references to Memory are checked to insure that they lie within the memory range allocated to that object program. In addition, a File Protect Register is optionally available that can protect any number of non-contiguous blocks of memory. The File Protect Register can be read or set only by the operating system.

C. PROGRAMMING FEATURES

The M-605 instruction repertoire is oriented toward the real-time application of the computer system and is a compatible subset of the larger 600-line systems. All members of the family use a common 36 bit, single address instruction with 18 bit half word, 36 bit single precision, and 72 bit double precision fixed and floating point operations.

With the common word size and instruction format, M-605 programs may be directly executed on larger 600-line systems. The 600-line floating point and double precision instructions not included in the M-605 repertoire are provided for on the M-605 through the use of subroutines. Macro operations are used to provide the linkage between the object program and the appropriate subroutine.

In addition to the usual repertoire found with most large-scale digital computers, the M-605 includes a number of instructions that facilitate real-time data processing: both 6 and 9 bit byte handling and binary-to-BCD instructions to facilitate character handling of both 6-bit BCD and extended character sets such as 8-bit ASCII; comparison instructions to permit the testing of tolerance and threshold values, magnitudes, or selected bits; and repeat instructions for iterating instructions in the repertoire.

The first repeat instruction permits the execution of one instruction a specified number of times with a specified address increment for operation on sequential lists. The second repeat instruction is used to search or operate upon a threaded list. The repeat instructions are particularly useful when used in conjunction with the comparison instructions.

The repertoire includes four basic methods of instruction address modification with 25 variations. These methods and variations are as follows:

- Register - Modify the address by one of the eight Index Registers provided, by the upper or lower half of either the A- or Q- Registers, by the Instruction Counter, or by use of the Direct Upper or Direct Lower operand variation. (16 variations, considering no modification as a variation.)
- Register then Indirect - Modify by Register (above) and then perform indirect addressing.
- Indirect then Register - Perform indirect addressing followed by Register modification (above).
- Indirect then Tally - Perform word or byte processing by indirect addressing with nine variations.

D. PROGRAMMING SYSTEM

The M-605 programming system provided by General Electric includes three classifications of programs:

- Operations Control: The General Comprehensive Operating Supervisor (GECOS/605) is provided as the operating supervisor for the M-605. Included in the operating system is the program loader, GELOAD/605, and RT-IOS, the real-time input/output supervisor.
- Language Processors: A FORTRAN IV compiler is provided with input to GMAP/605, the macro assembly program.
- A library of mathematical and service routines.

The above programs for execution on the M-605 are described in a later section of this manual.

In addition to the software provided for direct execution on the M-605, the common software interface used, GMAP source language, permits the advanced language of 600-line software to be used with the M-605. Program compilation using COBOL-61 Extended, ALGOL, or JOVIAL may be accomplished on any large scale 600-line system, and the resulting output in GMAP source language assembled for the M-605 substituting MACRO subroutine calls for instructions not in the M-605 repertoire.

E. INPUT/OUTPUT

Both standard peripheral devices and external real-time devices interface with the computer through the RT-IOC.

The M-605 peripheral subsystems are described in a later section of this manual. The peripheral complement is complete and compatible with the operating environment encountered with ground-based, mobile, and shipborne applications.

Of significant importance in real-time systems is the capability of the RT-IOC to accommodate real-time input/output of various transfer rates and formats. Each of the bi-directional RT-IOC channels can include a full set of command and address lines to permit the external device to completely control the I/O transaction if required. The RT-IOC design philosophy, of customizing its channel interface to each unique external device rather than requiring such devices to present a standard peripheral interface to the computer, provides for programming simplicity and reduced hardware complexity. The RT-IOC is fully described in the manual, "Real-Time Input/Output Controller Reference Manual."

II. COMPUTER FEATURES

A. BASIC MODULES

The computer consists of three independent modules; the Memory, the Processor, and the Real-Time Input/Output Controller (RT-IOC).

1. Memory

Unlike most computer systems which are processor-oriented, the M-605 Computer is memory-oriented with the Memory module serving as the center of the system. All information transfer and routing of control functions between Processor and peripheral devices or between Processors is through the Memory module.

The Memory module consists of the Memory Controller which performs all the logical and control functions and the Core Storage Unit which performs the storage function. The module is supplied with 16, 384 - 32, 768 - or 65, 536 - 36 bit plus parity words of core storage.

Although the single precision word size used throughout the system is 36 bits, each access to memory obtains a 72 bit double word with 2 bits for parity. In the case of single precision data, the least significant bit of the address specified by either a Processor or the RT-IOC selects the upper or lower half of the 72 bit double word.

Up to eight devices - either Processor or RT-IOC modules in any combination - can be connected to, and communicate with, each Memory module. Although Processor and RT-IOC modules are normally the only devices that communicate directly with the Memory module, non-standard devices may be connected to the Memory if required. Each of the eight devices that may be connected to the Memory operates asynchronously and makes requests of the Memory for access. The Memory Controller provides the function of servicing the demands for access in a priority order to permit the devices to time-share the Memory. No distinction is made between Processor and RT-IOC storage requests except that Processors are assigned a lower priority to avoid saturation of the Memory due to the high Processor demand rate.

In addition to the time sharing provisions, the Memory Controller includes a number of system control functions. These items which are discussed in later paragraphs are:

- Program interrupt facility. Up to 32 interrupt cells are provided which can be set by any of the eight devices connected to the Memory to initiate control Processor activity.
- Program-addressable interrupt mask. The control program can override the built-in priority of the interrupt cells and exercise complete control of all system interrupts as required.

- Program-addressable memory protect register. A register in the Memory Controller can be set to protect various multiple, non-contiguous blocks of memory as required. Each bit of the register corresponds to a storage block of 1024-37 bit words. This program-addressable memory protect register is a separate and independent feature that is unrelated to the memory protection provided through the Base Address Register in the Processor.

- Program-addressable memory port "lock-out" mask. A mask is provided to permit the program to mask any of the eight devices connected to Memory to inhibit data transfer and interrupts.

- Control-Processor designation facility. Provision is made in the Memory Controller to designate one of the several Processors that may be connected to the Memory as the "Control" Processor. Any interrupts set in the Memory will be sent to this Control Processor. Also, only that Processor may exercise control functions with the Memory module.

- Fault detection. Provision is made in the Memory for detection of certain faults including parity error, attempt to access a protected memory area, illegal attempt to exercise control functions, and others. Occurrence of a fault is brought to the attention of the device that was involved in the transaction during which the fault occurred.

2. Processor

The Processor module has full program execution capability and conducts all actual computational processing within the M-605 system.

The Processor can communicate with up to four Memory modules and, through an 18 bit address field, can directly address 262, 144-37 bit words of memory. Memory addresses are assigned consecutively beginning at zero and continuing through the full available memory. All Memory modules may appear to the Processor as a single memory with contiguous addresses.

a. DUAL MODE OPERATION

The Processor module is actually two Processors in one since it has two modes of operation: Master and Slave. The dual mode of operation is provided to insure a positive means of isolation of the operating system from object programs.

In the Master mode, no restrictions are imposed on a program. When in the Slave mode, a program is inhibited from exercising a control relationship with the Memory modules and input/output devices. Also, while in the Slave mode, the Base Address Register controls relative addressing and memory limits protection for the program.

Entry into the Master mode is caused automatically by the acknowledgement of interrupts and fault traps. Entry is also made when any object program executes a Master Mode Entry instruction. This type of Master mode entry is a "combination lock" that guarantees a legal entry into the Master mode and the operating system, thereby assuring complete system control. The Slave mode is set by the operating system when it initiates an object program by executing a Transfer and Set Slave Mode instruction.

b. ADDRESS TRANSLATION

To perform automatic program relocation, any program address to be used in a memory access request while the Processor is in the Slave mode is first translated into an actual address and then submitted to the memory.

For the purpose of address translation, the Processor Base Address Register contains a base address in bit positions 0-7. The translation takes place only in the Slave mode of operation. It consists of adding this base address to bit positions 0-7 of the program address.

In the Master mode no address translation takes place. Any program address to be used in a memory access request while the Processor is in the Master mode is used directly as an actual address and submitted to the memory without any translation.

Address translation is actually based on nine bits, the Base Address Register positions 0-8 and the bit positions 0-8 of the program address; this permits address relocation by multiples of 512 words. Because of a software requirement, bit position 8 of the Base Address Register has been wired to zero permanently and cannot be altered by the Load Base Address Register (LBAR) instruction. Thus, address relocation is performed in multiples of 1024. The BAR can be read or set only in the Master mode.

c. MEMORY PROTECTION

Any object program address to be used in a memory access request while the Processor is in the Slave mode is checked, just prior to fetch, to assure that it is within the address range allocated to the program for this execution. This address range protection is commonly referred to as memory protection.

For the purpose of memory protection, the 18-bit Processor Base Address Register is loaded with an address range in bit positions 9-16. The check takes place only in the Slave mode. It consists of subtracting bit positions 0-7 of the program address from this address range. When the result is zero or negative, the program address is out of range, and a Memory fault trap occurs.

More specifically, the checking is actually based on nine bits, the Base Address Register positions 9-17 and the bit positions 0-8 of the program address. This permits address range allocation to job programs in multiples of 512 words. Because of a software requirement, bit 17 of the Base Address Register has been wired to zero permanently and cannot be altered by the LBAR instruction. Thus, memory allocation and protection is performed in multiples of 1024 words.

In the Master mode no checking takes place; thus, any memory location (in those Memory modules that are connected to this Processor) can be accessed.

3. Real-Time Input/Output Controller (RT-IOC)

The RT-IOC is a general purpose I/O controller used to interface both standard peripheral equipments and non-standard devices to the computer.

Each RT-IOC has 27 external channels for communication with peripheral and external devices. These channels, which can be "expanded" to service multiple devices, can be tailored to the device with which the RT-IOC is to communicate. Data element sizes of 6, 9, 12, 18, or 36 bits can be accommodated.

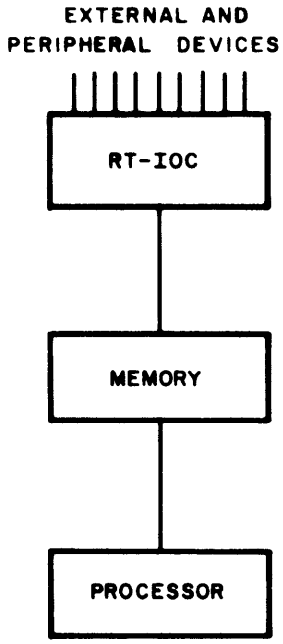
As with the Processor, the RT-IOC can communicate with up to four Memory modules; thus, each peripheral device has access to a maximum of 262, 144 words of storage.

As optional features, a program loader for automatic program reload is available. Also, parity generation and checking is available as an option. Such checking or generation is accomplished for all data transfers between the RT-IOC and the external device.

B. CONFIGURATIONS

1. Basic

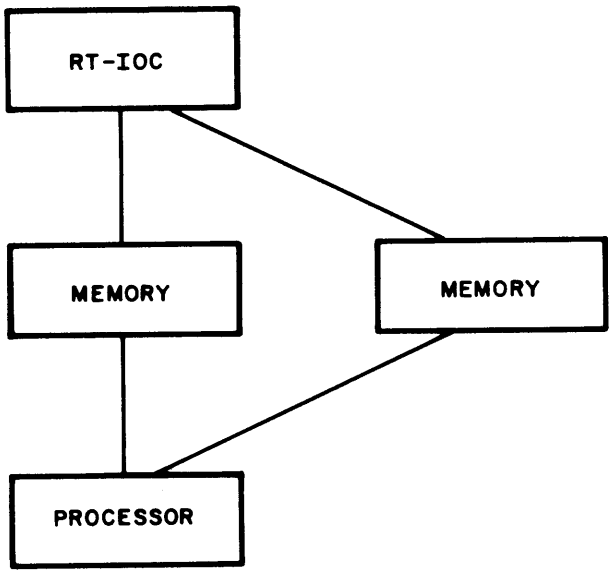
The simplest configuration of the M-605 uses one Memory, one Processor, and one RT-IOC module as shown below.



From the simplest configuration, expansion can be made in various directions. To accommodate additional I/O devices, for example, additional RT-IOC modules can be cabled to any of the unused Memory ports. For non-standard devices that have an extremely high data rate approaching the maximum capability of the Memory, it is possible to interface such devices directly to a Memory port.

2. Multi-Memory

The addition of one or more Memory modules to the basic configuration may be made to increase the available on-line core storage. As shown below, a second Processor and RT-IOC port is used to communicate with the second Memory module.

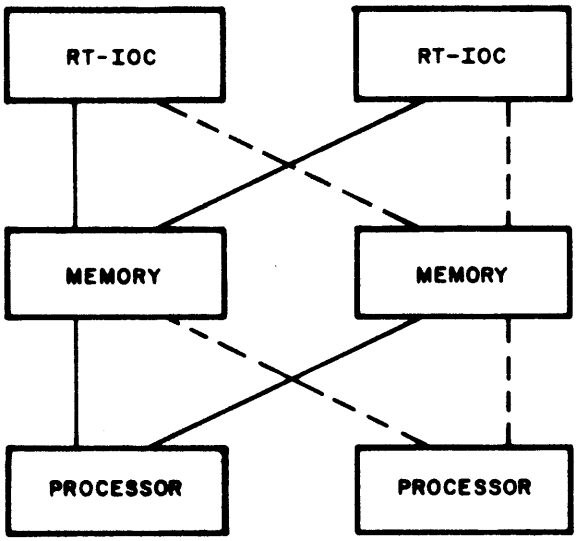


The use of multiple Memory modules also increases system performance over that obtained with the simplest configuration. With one Memory, both the I/O devices and the Processor time-share the available memory accesses. This results in program execution times greater than if no I/O transfer took place. With several Memory modules, the I/O load may be distributed over the two Memory modules with the resulting improvement in program execution time.

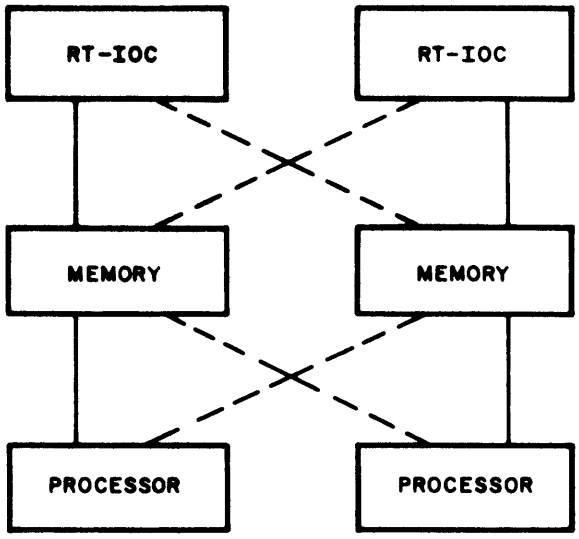
3. Multi-Processor

Additional Processor modules may be used to increase the computational capability of the system. A configuration using two of each of the primary modules is shown below.

The configuration shown is a multi-processor configuration in which only one of the several Processors that may be in the system is designated as the "Control" Processor. All other Processors are subservient to the Control Processor. Subservient Processors can perform processing under the direction of the Control Processor; however, they cannot control system operations nor can they directly initiate input/output transfers.



4. Multi-Computer



In the multi-computer configuration, there are two or more Processors designated as Control Processors. In this configuration, each Processor controls its own Memory modules and initiates input/output operations through its own RT-IOC. A system of this type can be considered as two or more independent computer systems. In applications where automatic failure detection and recovery is necessary, either system can serve as backup to the other. One system may be on-line performing the primary real-time mission while the other is off-line performing routine data processing functions yet continuously available for recall to on-line status if required.

5. Control Processor Designation

The type of configuration is solely specified by the control processor designation facility in each Memory module and may be readily changed. Of the several Processors that may be connected to a given Memory module, only one is specified as the "Control" Processor for that Memory. Any interrupts received by the Memory module will be sent to the Control Processor for that Memory. Similarly, only the Control Processor can set or read the Mask and File Protect Registers in that Memory and issue CIOC instructions to the peripheral devices associated with that Memory.

In the multi-processor configuration, therefore, all interrupts received in any of the Memory modules will interrupt only the Control Processor. When subservient Processors wish to interrupt the Control Processor to notify it of task completion or request that some action be performed by the operating system, they may do so by setting an interrupt cell in any of the Memory modules.

To permit a Control Processor to interrupt a subservient Processor to initiate some activity, provision is made whereby the Control Processor can issue an instruction (CIOC) that causes a psuedo-fault in the specified subservient Processor. The occurrence of the fault will force the execution of an instruction pair by the subservient Processor resulting in a transfer.

6. Reconfiguration

In either the multi-processor or multi-computer configuration, the provision to reconfigure the system - to substitute modules or groups of modules without a corresponding change in the program - is of major importance.

When either a Processor or an RT-IOC communicates with more than one Memory, the high-order bits of the address designate the Memory module. To permit an exchange of Memory modules for purposes of reconfiguration, provision is made in the Processor and RT-IOC to alter the interpretation of the high-order address bits. The effect of the alteration is to reroute the access request to a corresponding location in another Memory module. Thus, in the event a Memory is out of service for any reason, such as maintenance or repair, substitution can be made without a program change. The reconfiguration can be manually effected from the maintenance panels associated with each device or automatically in applications requiring automatic failure detection and recovery.

C. INTERRUPTS AND FAULTS

1. Program Interrupts

The ability of external devices or events to interrupt the program being executed by a Processor is an essential factor in the multi-programming ability of the M-605 in both real-time and non-real-time environments.

In real-time applications, the system in which the computer complex is used must be able to control, or time-share, the execution of all programs of a given program set to allow the complex to respond with a minimum of delay to system requests for various computations or controls.

In both real-time and non-real-time applications, peripheral devices must be able to interrupt the program being executed to signal the occurrence of an external event.

Located in each Memory module is a program interrupt facility that consists of up to 32 unique interrupt cells. Although any of the eight devices - either Processor or RT-IOC modules - that may be connected to the Memory module can set any of the cells, only specific cells will generally be assigned to a given device. Associated with the 32 cells is a 32 bit Mask Register that is read or set by program and which can be used to change the wired priority of the 32 cells. A binary 1 or 0 in a given bit position of the Mask Register will respectively permit or inhibit the acknowledgement of interrupt requests made by one of the devices connected to the Memory. If a device requests a program interrupt by setting one of the cells and if the cell is unmasked, the interrupt will be acknowledged. (If several demands are made simultaneously, the highest in priority will be serviced first.)

Whenever an unmasked interrupt cell has been set, the Memory presents an "interrupt present" flag to the Processor designated as the Control Processor. As soon as the Processor has completed the current instruction and assuming that interruption has not been inhibited, the Processor will interrupt its program sequence and request of the Memory the number of the cell causing the interrupt. Using this number as a part of an address, the Processor executes the pair of instructions at one of 32 core locations corresponding to the 32 interrupt cells. These instructions can transfer program control to the entry point of the desired routine which can safe store the Processor's Instruction Counter and registers to permit a later return to the interrupted program.

The Mask Register is used to change the priority. Once a program is initiated, the Mask Register is set to permit interruption of the program only by events of a higher priority than the one that initiated the current program.

Although interrupts commonly cause a transfer of control to the operating system, the transfer can be direct to a specific program that is to respond to the interrupt without the intervention of an executive program to determine the priority of the interrupt. This is significant in real-time applications to minimize the computer response time.

The 32 core locations associated with the 32 interrupt cells, are located in the low order Memory module. If a Processor has a control relationship with more than one Memory module, each block of 32 locations, one block per Memory module, is located in the low order Memory module in contiguous locations.

A program may inhibit interruption by placing a binary 1 in bit position 28 of an instruction. When specified, interruption is inhibited until the execution of an instruction that does not inhibit interruption. Certain instructions always prevent interruption following their execution.

Unless the instruction pair executed in response to the interrupt specifies differently, the Processor will enter the Master mode of operation.

2. Faults

Because of the continuous access needs in a real-time application, and in keeping with the interrupt-oriented philosophy of the M-605, the Processor provides for continuous on-line operation by responding to high level interrupts called faults. Any operation that might otherwise cause the system to "hang-up" causes a trap to a fault routine so that remedial action may be taken.

As with interrupts, a pair of instructions are executed as a response to a fault. Sixteen instruction pairs each corresponding to one of the sixteen faults are located in a block of storage.

There are four general categories of faults:

- Instruction generated
- Program generated
- Hardware generated
- Manually generated

Not all the sixteen are actually faults but are treated the same as faults.

a. INSTRUCTION GENERATED FAULTS. The Instruction generated faults are:

(1) Master Mode Entry (MME)

The instruction Master Mode Entry has been executed.

(2) Derail (DRL)

The instruction Derail has been executed.

(3) Fault Tag

The address modifier I(T) where T=F has been recognized. The indirect cycle will not be made upon recognition of F, nor will the operation be completed.

(4) Connect (CON)

The Processor has received a Connect from a Control Processor via a Memory module.

(5) Illegal OP Code (ZOP)

An operation code of all zeros has been executed.

b. PROGRAM GENERATED FAULTS. Program generated faults are defined as:

(1) Arithmetic Faults

- Overflow (FOFL) -- An arithmetic overflow, exponent overflow, or exponent underflow has been generated.
- Divide Check (FDIV) -- A divide check fault occurs when the actual division cannot be carried out for one of the reasons specified with each divide instruction.

(2) Elapsed Time Interval Faults

- Timer Runout (TROF) -- This fault is generated when the timer count reaches zero. If the Processor is in Master mode, recognition of this fault will be delayed until the Processor returns to the Slave mode; this delay does not inhibit the counting in the Timer Register.
- Lockup (LUF) -- The Processor is in a program lockup which inhibits recognizing an execute interrupt or interrupt type fault for greater than 16 milliseconds. Examples of this condition are the coding TRA* or the continuous use of the inhibit bit.
- Operation Not Completed (FONC) -- This fault is generated due to one of the following:

  No Memory module attached to the Processor for the address.

  Operation Not Completed.

(3) Memory Faults

- Command (FCMD) -- This fault is interpreted as an illegal request by the Processor for action of the memory. These illegal requests are:

  The Processor is not the Control Processor, or is the Control Processor in the Slave mode, and issues a CIOC, RMCM, RMFP, SMCM, SMFP, or SMIC.

  The Processor has issued a connect to a channel that is masked off (by program or switch).

- Memory (FMEM) -- This fault is generated when:

  No physical memory existed for the address.

  An address (in Slave mode) is outside the program boundary or protected memory.

  The memory did not respond to a request within 1 to 2 milliseconds.

c. HARDWARE-GENERATED FAULTS. The hardware-generated faults are defined as:

(1) Operation Not Completed (FONC) -- This fault is generated due to one of the following:

The Processor has not generated a memory operation within 1 to 2 milliseconds and is not executing the Delay Until Interrupt Signal (DIS) instruction.

The memory closed out a double-precision or read-alter-rewrite cycle.

(2) Parity (FPAR) -- This fault is generated when a parity error exists in a word which is read from a core location.

d. MANUALLY GENERATED FAULTS. Manually generated faults are:

(1) Execute (EXF)

- The EXECUTE pushbutton on the Processor maintenance panel has been activated.
- An external frequency has been substituted for the EXECUTE pushbutton.

The above two faults are dependent on other switch positions on the Processor control panel.

(2) The Power Turn On/Off Faults

- Startup (SUF) -- A power turn-on has occurred.
- Shutdown (SDF) -- Power will be turned off in approximately 1 millisecond.

III. PROGRAMMING

A. WORD FORMATS

1. Fixed-Point Data

Fixed-point operations are conducted with 36-bit, single-precision operands; 72-bit double-precision operands; or 18-bit, half-word operands. The formats of the three types of operands are shown in Figure III-1.

Fixed-point data is represented in the two's complement number system. The range of single-precision numbers is $2^{35}-1$ to -2^{35}, the range of double-precision numbers is $2^{71}-1$ to -2^{71}, and the range of half-word operands is $2^{17}-1$ to -2^{17}.

2. Floating-Point Data

Floating-point operations are conducted with either 36-bit, single-precision operands or 72-bit, double-precision operands. The formats of each are illustrated in Figure III-2.

Floating-point numbers are represented by a mantissa of 28 or 64 bits and a binary exponent of 8 bits. Both the exponent and the mantissa are represented in the two's complement number system. The first bit of the mantissa indicates the sign of the quantity, and the exponent has a range of +127 to -128.

Although the M-605 hardware does not include floating point instructions, the above floating point data format is assumed by floating point subroutines.

3. Alphanumeric Data

Either six 6-bit characters or four 9-bit characters are contained in each data word. The designation of the characters within the word is as shown in Figure III-3. Any combination of characters can be read or written to or from memory.

B. INSTRUCTION FORMAT

With the exception of Repeat instructions and the character-handling instructions, each instruction has the format shown in Figure III-4.

The operation (OP) part of the instruction specifies the operation to be performed and the registers that are involved.

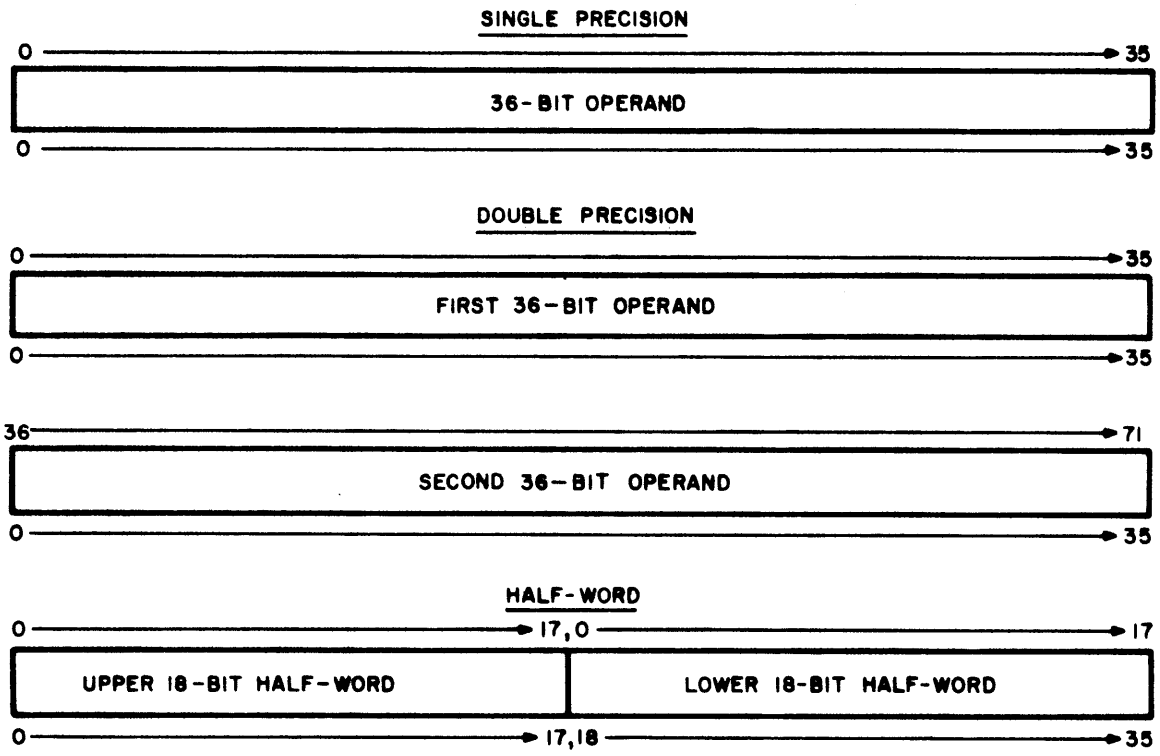


Figure III-1. Fixed Point Data Formats

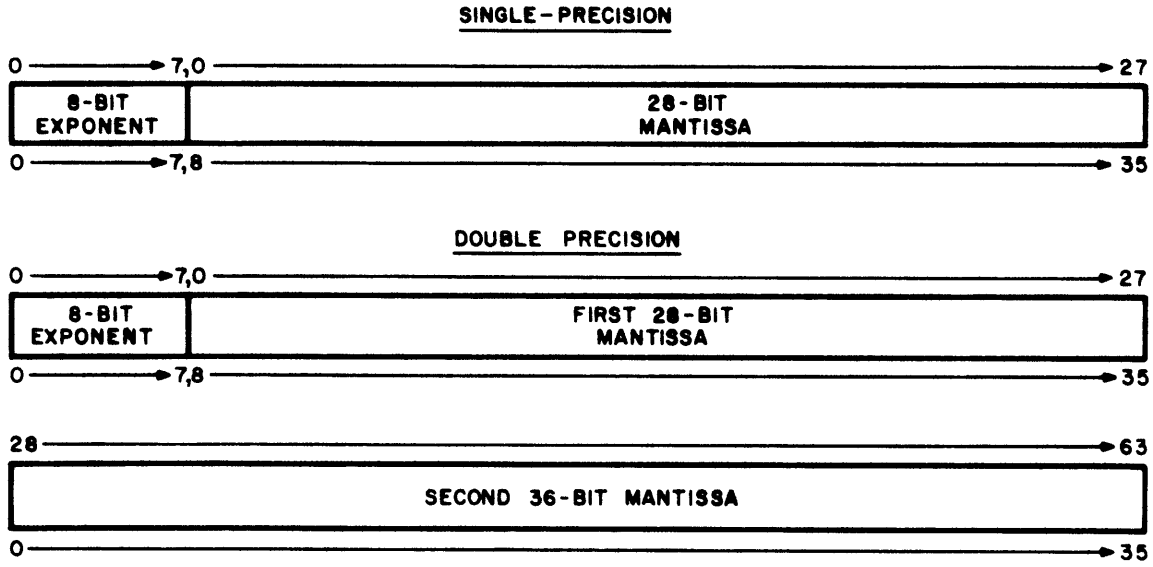


Figure III-2. Floating-Point Data Formats

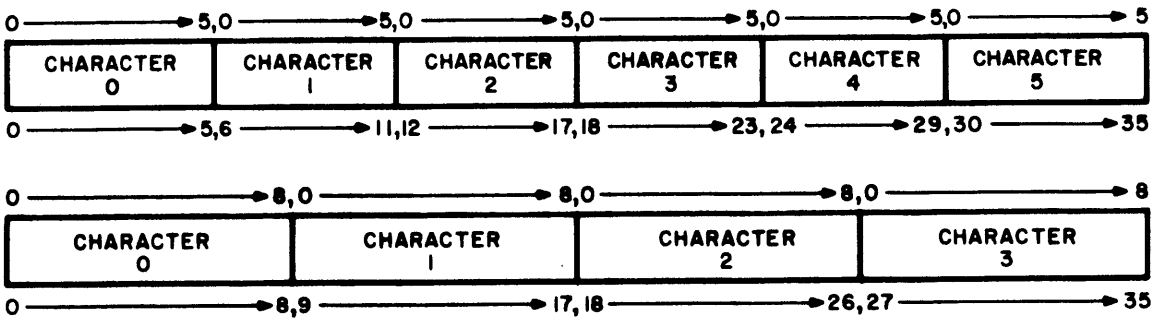


Figure III-3. Alphanumeric Data Formats

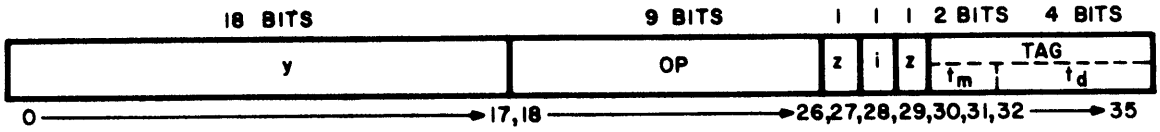


Figure III-4. General Instruction Format

The address (y) field of the instruction word specifies the storage location to obtain an operand or to place the result of the specified operation. Some of the exceptions to this general rule are:

- Shift instruction, where the address field designates the number of bit positions to shift
- Program sequence transfers, where the address field designates the location of the next instruction to be executed
- A group of instructions where the address field specifies sub-operations.

The address modifier (t_m) portion of the tag field specifies how the address contained in the instruction word is to be modified to form the effective address Y of the resultant or the operand.

The designator (t_d) portion of the tag field specifies the type of register modification to be used.

The interrupt inhibit (i) portion of the instruction word controls program interrupts in the Processor by either fault or program execute interrupts from the Memory.

The (z) fields must be zero, and are reserved for future compatibility with other General Electric 600-line systems.

C. REGISTER DESCRIPTIONS

The internal Processor registers that are accessible to the program are:

Name	Mnemonic	Length
Combined A-Q Register	AQ	72 bits
Eight Index Registers (n = 0, 1, ..., 7)	Xn	18 bits each
Exponent Register	E	8 bits
Base Address Register	BAR	18 bits
Indicator Register	IR	18 bits
Timer Register	TR	24 bits
Instruction Counter	IC	18 bits

- The AQ-Register can be used:

  In floating-point operations as a mantissa register for both single and double precision;

  In fixed-point operations as an operand register for double precision, and each half independently of the other as two operand registers for single precision; the halves become the A-Register (bits 0 through 35) and the Q-Register (bits 36 through 71); and

  In address modification each half of the A-Register and of the Q-Register can be the source of an index; these halves then become AU (bits 0 through 17), AL (bits 18 through 35), QU (bits 0 through 17), and QL (bits 18 through 35).

- The Xn-Registers can be used:

  In fixed-point operations as operand registers for half-precision;

  In address modification as sources of index quantities.

- The E-Register supplements the AQ-Register in floating-point operations as exponent register.

- The Base Address Register is used in address translation and protection. It denotes the base address and the number of 1024-word blocks assigned to the program being executed.

- The Indicator Register is a generic term for all of the program-accessible indicators within the Processor; the name is used where the set of indicators appears as a register, that is, as a source or destination of data.

- The Timer Register is decremented by one each 16 microseconds and a Timer Runout fault occurs whenever its contents reach zero. If Timer Runout occurs in the Master mode, the fault does not occur until the Processor returns to the Slave mode.

- The Instruction Counter stores the address of the next instruction to be executed.

D. ADDRESS MODIFICATION

1. Modification Types

Address modification can be performed in four basic ways, using the identifiers R, RI, IR, and IT. The modification types R, RI, IR, or IT are specified by unique binary codes placed in the t_m field

of the instruction word. The registers used to modify R, RI, or IR addresses are indicated in the t_d field of an instruction or indirect word.

- Register modification (R) -- Modify the address by adding to it the contents of the indicated register, producing the effective address of the operand. Indexing registers are X0-X7, AU, AL, QU, QL, and IC. The direct operand modifier symbols DU and DL may be used to treat the address directly as the operand.

- Register then Indirect (RI) -- First perform the indicated register modification of the address, as with R; then obtain an indirect word and conduct the modification specified in the indirect word.

- Indirect then Register (IR) -- First obtain the indirect word from the original address; then conduct the modification specified in the indirect word. Upon completion of the indirect addressing, perform the indicated register modification in the last IR word encountered. (Some special conditions for modification may be encountered in the indirect words; these involve R, RI, and IT.)

- Indirect then Tally (IT) -- Obtain the indirect word using the address modification type IT as specified in the t_m field of the instruction; then use the new address field of the indirect word to get the effective operand address. Next, follow one of nine possible variations, indicated in the t_d field of the instruction or indirect word that specified IT. (Two of the nine variations (SC and CI) are used for handling characters.)

The nine possible variations of the IT modification are listed and described below. When used, each variation mnemonic appears as a unique bit configuration in the t_d field of its instruction.

VARIATION SUMMARY UNDER IT MODIFICATION

Mnemonic	Name
(t_d) = I	Indirect (unmodified)
= ID	Increment address, decrement tally
= DI	Decrement address, increment tally
= SC	Sequence character
= CI	Character from indirect
= AD	Add delta
= F	Fault
= IDC	Increment address, decrement tally, and continue
= DIC	Decrement address, increment tally, and continue

IT Variation Descriptions

I — This tag is used to reference an indirect word without disturbing any part of the word. See Figure III-5.

Figure III-5. Indirect Word as Interpreted Under I

ID This tag is used to reference an indirect word (Figure III-6) and to increment the address by 1 and decrement the tally of the indirect word by 1.

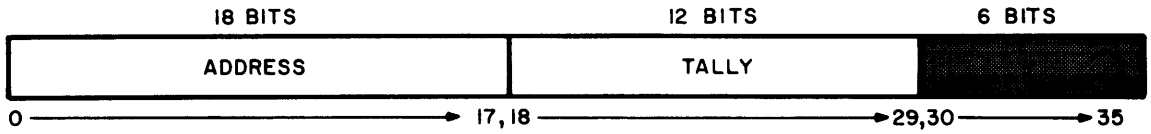


Figure III-6. Indirect Word as Interpreted under ID or DI

DI This tag is used to reference an indirect word and to decrement the address by 1 and increment the tally of the indirect word by 1.

SC This tag is used to reference an indirect word (Figure III-7) and to decrement the tally by 1, increment the character position by 1, and to specify the next character position--all in the indirect word. C specifies either 6 or 9 bit characters.

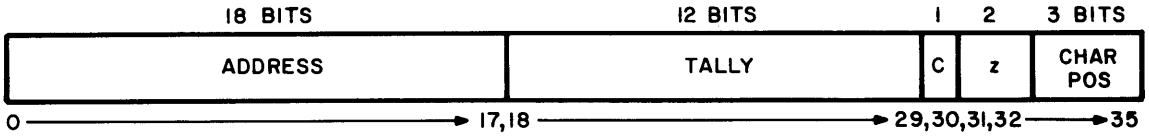


Figure III-7. Indirect Word as Interpreted under SC

CI This tag is used to reference an indirect word (Figure III-8) and to operate on a particular character position without disturbing the indirect word. C specifies either 6 or 9 bit characters.

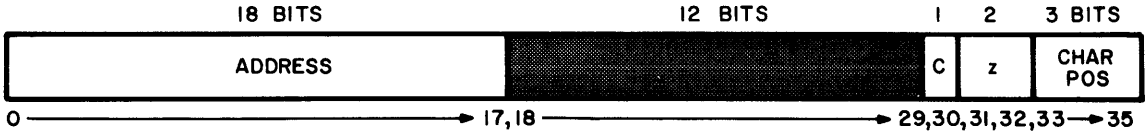


Figure III-8. Indirect Word as Interpreted under CI

AD This tag is used to reference an indirect word (Figure III-9) and to increase the address field by the figure in the delta portion of the instruction. The tally portion of this indirect word is decremented by 1.

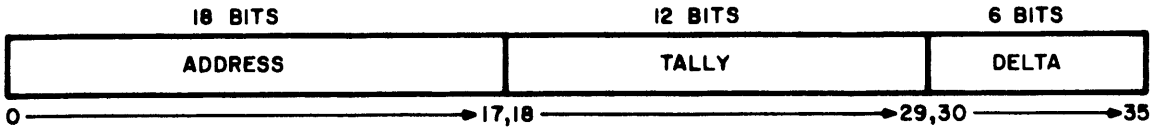


Figure III-9. Indirect Word as Interpreted under AD

F — The use of this address modification will cause a fault interrupt to occur.

IDC — This instruction modification is similar to ID modification, except that t_m and t_d fields specify further address modification. See Figure III-10 for indirect word interpretation.

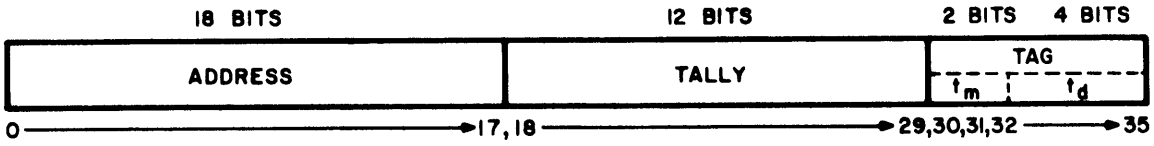


Figure III-10. Indirect Word as Interpreted under IDC and DIC

DIC — This tag modification is similar to DI modification, except that the t_m and t_d fields specify further address modification.

Address modification using registers takes zero time; that is, under R, RI, IR modification, indexing the operand address of an instruction or indirect word adds no time to instruction execution. Under R modification, the DU and DL variations reduce normal execution time.

2. Character Operations

Operations involving characters are performed using instructions that specify the IT type of address modification in the t_m field. The I of IT obtains an indirect word that, in turn, references a location containing the character or characters to be used. The indirect word holds tallying and character identifying information while the referenced location holds the character data. The t_d field of the original instruction must contain the sequence character (SC) or character from indirect (CI) designator variations.

Each time the original instruction is used under the SC variation, the tally and character position information in the indirect word are automatically changed by one to prepare for operating on the next sequential character and for terminating the SC operations when the tally reaches zero. After operating on all characters in the core location the Processor automatically increases the data word address by one and starts again at character 0 in the new location.

When a single character within a word is to be used repeatedly with the IT modification, the t_d field designates CI. The indirect word format is that shown in Figure III-8. The data movement process parallels that for the SC variation but the indirect word is not altered.

For both variations, the character position field of the indirect word (bits 33 through 35) specifies in octal the character position of the memory location to be used in instruction execution. Figure III-11 illustrates typical single-character operations to and from memory, assuming 6-bit characters and a character position designation of 3.

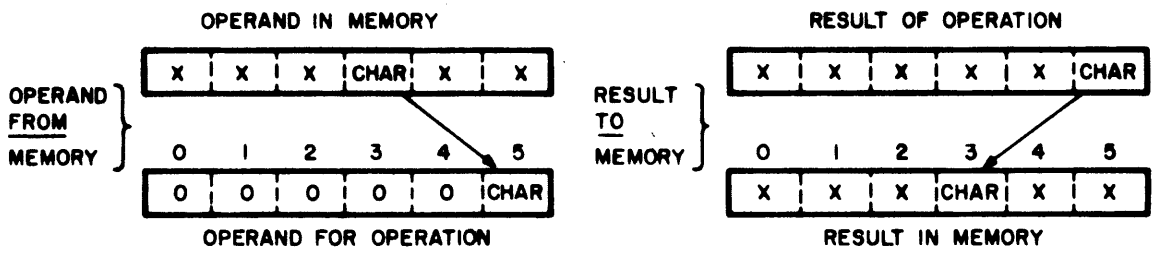


Figure III-11. Typical Character Operations

For operations where the operand is taken from memory, the effective operand is presented as a single word with the specified character right-justified to character position 5. The remaining positions (0 through 4 in Figure III-11) are presented as zeros.

For operations where the resultant is placed in memory, character 5 of the resultant replaces the specified character (3 in Figure III-11) in memory and the remaining character positions are not affected.

E. INSTRUCTION REPERTOIRE

Most of the instructions are familiar to experienced programmers of large-scale computers. However, additional instructions have been provided to give the programmer extended capability for character handling, decision-making, and advanced programming techniques involving list processing.

The basic instructions have provision for multiple variations by permitting the programmer to specify not only the type of address modification desired, but also the source registers associated with particular operation codes. For example, the operation field for a Transfer and Save Instruction Counter in Index instruction specifies the index in the operation field, leaving full address modification capability free for destination calculation.

A Processor module permits efficient operations to be performed on 18- and 36-bit operands. The 18-bit operands are stored with, and fetched as, the left half of the instruction.

The following paragraphs briefly describe salient features of the major instruction types. A complete listing of the full instruction repertoire and execution times is included in the Appendix.

1. Data Movement

Character handling and manipulation is facilitated by the "indirect and tally" indexing option, and by instructions for directly loading and storing selected characters of the A- or Q-Registers.

Instructions are also included for directly loading the index registers from either memory or the A- and Q-Registers, directly storing any register into memory, and for loading registers with the complement of the memory location specified.

2. Shifting

The A- and Q-Registers can be shifted individually or as one unit. The shift commands include right or left shift arithmetic, right shift logical, and left shift rotate.

3. Fixed-Point Arithmetic

Fractional and integer instructions for both multiplication and division afford the programmer freedom from scaling the results of such operations. Fractional multiplications are performed with the multiplicand in the A-Register; the result appears in bit positions 0 through 70 of the AQ-Register, automatically scaled with the binary point to the left of position 0. Integer multiplications are performed with the multiplicand in the Q-Register; the result appears in bit positions 1 through 71 of the AQ-Register, automatically scaled with the binary point to the right of position 71.

Fractional divisions use the full range of the AQ -Register for the dividend; the quotient appears in the A-Register with the remainder in the Q-Register. The binary point is automatically scaled to the left of position 0. Integer divisions have the integer dividend in the Q-Register, with the binary point to the right of position 35. After division, the quotient is in the Q-Register with the binary point automatically placed to the right of position 35, and the remainder is in the A-Register.

Normally, integer operations of divide and multiply occur in the Q-Register, and fractional operations of divide and multiply occur in the A-Register. This convention permits easy programming of fixed-point arithmetic operations.

Two arithmetic and three logic instructions used frequently in program coding are provided for combining the contents of memory locations directly with the contents of registers and storing the results in the same locations, without recourse to separate Store instructions. In all cases, the programmer can use the 18-bit indexing registers, X0 through X7, and the 36-bit A- and Q-Registers. In effect, the Add and Subtract to Storage instructions make arithmetic accumulators of all available memory locations. In all cases, the register contents are undisturbed.

4. Boolean Operations

The logical operations AND, Exclusive OR, and Inclusive OR are permitted between storage and the Xn-Registers, and the A- and Q-Registers.

5. Comparison

Compare operations do not alter the contents of storage or the specified register, but merely set or clear the appropriate indicator as the result dictates. The compare instructions enable the programmer to make many types of program decisions.

The fixed-point compare instructions are:

Instruction	Registers Used	Principle Functions
Compare Magnitude	A	Compare absolute values
Compare with Register	A, Q, X0-X7	(1) Compares algebraic values (2) Compares characters
Comparative AND with Register	A, Q, X0-X7	Tests for all zeros in word fields
Comparative Not-AND with Register	A, Q, X0-X7	Tests for all ones in word fields
Compare Masked	A	Searches for identical, selectable word fields
Compare with Limits	A, Q	Searches for a value within selectable limits.

Floating-point compare instructions are included for single- and double-precision operations on absolute values and algebraic values. All compare instructions are repeatable using the RPT, or RPL instructions described later under the heading, Special Operations.

6. Floating-Point Arithmetic

Floating point operations are performed by floating point subroutines. Operations can be performed on both single- and double-precision data words; complete sets of data movement, arithmetic, and control instructions are provided for use in both types of operations. Unless otherwise specified by the programmer, the mantissas of all floating-point operation results are automatically normalized. In additions and subtractions, addends and subtrahends are automatically aligned.

Operations on floating-point numbers are performed using an extended register composed of a 72-bit AQ-Register, which holds the mantissa, and a separate 8-bit exponent register. The existence of separate exponent and mantissa registers enables the programmer to intermix efficiently single- and double-precision instructions.

The floating-point instruction repertoire includes two especially convenient divide instructions: Floating Divide Inverted (FDI) and Double-Precision Floating Divide Inverted (DFDI). These instructions cause the contents of the memory location to be divided by the contents of the AQ-Register--

the reciprocal of other divide instructions in the repertoire. Thus, regardless of whether the contents of the AQ-Register must be a dividend or a divisor, the programmer can always perform a division without recourse to wasteful data movement operations.

Floating Negate, Normalize, Add to Exponent, and Single and Double Precision Compare Instructions further facilitate effective programming.

7. Transfer of Control

The complement of program transfer instructions permits the storage of the instruction counter in indexing registers X0 through X7 (at programmer option), an unconditional transfer, and conditional transfers. Conditional transfers on zero, plus, and carry have corollary transfers: non-zero, minus, and no carry. Transfers on overflows and underflows are to maskable fault routines. If the normal fault routine is masked, transfer is at programmer option.

8. Special Operations

Several special instructions are provided to expand programmer options and to reduce coding work through utilization of hardware features.

Two repeat instructions in the repertoire provide unusual programming advantages: Repeat (RPT), and Repeat Link (RPL). The RPT instruction permits execution of the next instruction a selected number of times according to program requirements; is especially useful for operating upon sequential lists in memory. For example, if RPT is used with any of several compare instructions to search a list, termination of the repeats will occur when a "hit" is made, according to programmer option as established by conditions in the RPT instruction. The "hit" causes transfer to the next sequential instruction.

The RPL instruction is similar in its execution to the RPT; it facilitates the processing of threaded lists scattered through memory.

The Effective Address to Register (EAR) instructions permit the effective address of such an instruction to be placed in any of the index registers, in the A-Register, or in the Q-Register. Thus, any effective address referenced frequently in a program can be stored in a register and used without lost processing time in repeatedly redeveloping the effective address. Furthermore, the EAR instructions provide the programmer with the ability to transfer data among any of the index registers, the A-Register, and the Q-Register.

The Binary to Binary Coded Decimal (BCD) instruction converts the magnitude of a 36-bit or smaller binary number to its decimal equivalent in BCD form. The conversion is made automatically, one decimal digit per instruction execution, using previously-stored conversion constants. The BCD form of the converted number is readily available for further operations.

The Gray to Binary (GTB) instruction converts a 36-bit word containing data in the Gray code (for example, coded analog information from an analog-to-digital input device) to its binary equivalent in only one execution of the instruction.

The Execute Single and Execute Double (XEC and XED) instructions allow the programmer to execute remote instructions singly or in pairs. Because the address counter is not disturbed by the execute remote instructions, the program will continue sequentially after executing the XEC or XED referenced instructions, providing the referenced instructions do not alter the address counter. If a referenced instruction affects the address counter, a program transfer occurs.

IV. PROGRAMMING SYSTEM

The standard software supplied with the M-605 Computer has been designed to both support the M-605 hardware in the performance of real-time missions and to provide support of scientific data processing. The M-605 standard software is compatible with equivalent software for the GE-625 and GE-635 Computers. The following standard software packages are provided:

GECOS/605	Executive Program
GMAP	Assembly Program
FORTRAN IV	Compiler
GELOAD/605	Loader

Utility - Library Routines

Test and Diagnostic Package

Support Software

A. GECOS/605

GECOS is the General Comprehensive Operating Supervisor. The interface of this supervisor for the M-605 Computer is compatible with the GECOS for larger-scale 600-line computers. Users of the system employ the same techniques when communicating with the operating system.

GECOS/605 will perform the following basic functions:

- Program loading
- Input-output supervision of all RT-IOC devices.
- Processing of multiplexed execute interrupts from the RT-IOC
- Standard fault processing
- Queuing of input-output requests
- Job sequencing

B. GMAP

GMAP is the macro-assembly program and accepts as input, symbolic machine language instructions and produces relocatable or absolute binary card images on the program output device (card punch, or magnetic tape). A list of the symbolic program and its assembled equivalent (in octal code) is produced on the output unit (printer, typewriter, or magnetic tape).

Language compilers translate source language statements into the GMAP language. GMAP provides the translation medium by which the symbolic code that is generated by the various language compilers is assembled into object code.

GMAP provides some of the conveniences of a compiler along with the flexibility of an assembler. The ability to design desired MACROS in order to provide convenient shorthand notations, plus the use of all 600-line machine instructions, as well as a complete set of pseudo-operations, provides the programmer with a very powerful and flexible tool.

The assembler is implemented to allow for updating and/or merging of an ALTER package to a previously prepared assembly input.

GMAP recognizes a full range of pseudo-operations for such items as data generation, symbol definition, storage allocation, program linkage, control and MACRO prototype generation. The MACRO operation provides a means of easily handling both the coding of a repeated pattern of instructions that within themselves contain variable entries at each iteration of the pattern, and basic coding patterns subject to conditional assembly at each occurrence.

C. FORTRAN IV

The FORTRAN IV compiler accepts the newest and most powerful features of the FORTRAN language. The source program written in FORTRAN IV language is translated into assembly language by the compiler which uses the unique instruction repertoire of the M-605 to produce highly-efficient object programs, at the same time providing fast compilation.

Some of the newest features of FORTRAN IV that are embodied in the M-605 FORTRAN compiler are:

- The ability to have variable-field input, utilizing the NAMELIST statement.
- Variable-field output
- Compile DEBUG statements in the object program.

The compiler also contains extensive language diagnostic checking features. The search for source program errors always proceeds to the end of the program in order to find all detectable errors.

Since machine language subroutines may be called directly by a FORTRAN program, real-time devices, (display, plotters, etc.) may be indirectly referenced in the language. These subroutines called by a FORTRAN program would request GECOS/605 to perform the input-output function. This concept of centralized input/output provides complete protection for the real-time mission.

D. GELOAD/605

The General Loader is a module of GECOS/605 and is used to load object programs for execution. GELOAD/605 will load relocatable or absolute binary card images produced by GMAP. The card images can be loaded from the card reader, magnetic tape or a mass storage unit. Independently assembled subroutines are linked by GELOAD/605 into one continuous program to make them function as a complete program. The programs are relocated in memory; however, the relative addresses of the program are not made absolute until execution time. This is accomplished automatically by the M-605 Processor module.

E. UTILITY—LIBRARY ROUTINES

A basic set of mathematical subroutines are available for the M-605 Computer. This library includes the following programs:

- Sine and cosine
- Exponential
- Logarithm
- Arctangent
- Hyperbolic tangent
- Square root
- Natural exponential
- Natural logarithm
- Arcsine and arccosine.

F. TEST AND DIAGNOSTIC PROGRAMS

Programs to test the modules and peripheral devices of the M-605 Computer are part of the standard software offering.

G. SUPPORT SOFTWARE

Since the 600-line software, in general, translates their source language statements into macro assembler (GMAP) language and since any GMAP program can be assembled for execution on the M-605 Computer, the advanced language processors of the GE-635/625 can be used to prepare programs for the M-605. These compilers include COBOL 61-Extended, ALGOL, and JOVIAL.

An assembler to prepare programs for execution on the M-605 Computer is available for the GE-225 and IBM7090/94 Computers. An M-605 simulator program for execution on the 7094 is also available.

V. PERIPHERAL EQUIPMENT SUBSYSTEMS

The following is a description of the peripheral subsystems included in the M-605 system.

All peripheral subsystems interface with the M-605 through the RT-IOC.

A. CONSOLES

Either free-standing or desk top consoles with an I/O typewriter are used with the system. In either case, the console is treated as a peripheral device and communicates through the RT-IOC. Included with the ten character per second typewriter are the following controls and displays:

AUTOMATIC PROGRAM RELOAD switch indicator

REQUEST and switch

ON-LINE/OFF-LINE indicator and switch

SEND/RECEIVE indicator

OPERATOR ATTENTION indicator

B. MAGNETIC TAPE SUBSYSTEM

The Magnetic Tape Subsystem consists of the Magnetic Tape Controller and the Tape Transport.

The transport has selectable densities of 200, 556, and 800 bits per inch. The tape passing speed is either 150 or 75 inches per second with a maximum transfer rate of 120,000 characters per second. The data format used is IBM 729 VI compatible.

The controller can operate with up to eight transports and provides full scatter-gather capability. Either program or operator density selection is provided.

C. CARD READER SUBSYSTEM

The Card Reader is an 1100 card per minute column reader. The format used is either standard Hollerith card code or column binary or intermixed using the 7-9 punch convention. The reader provides for continuous operation while loading or removing cards and has a 2000-card hopper and stacker capacity. As an option, a 250 card reject hopper is available.

D. CARD PUNCH SUBSYSTEM

The Card Punch is a 450 card per minute column punch. The format used is either standard Hollerith, edited Hollerith, or 12-row binary. The Punch provides for continued operation while loading or removing cards, and has a 2000 card hopper and stacker capacity.

E. LINE PRINTER SUBSYSTEM

Both 600 and 1000 lpm printer subsystems are provided. Either subsystem is a fully buffered printer with 132 columns and 64 printable characters. A Vertical Format Unit is included. The 600 and 1000 lpm printing rate is for the 46 most used characters. The rate using all 64 characters is 480 and 800 lpm for the 600 and 1000 lpm printers respectively.

F. PAPER TAPE SUBSYSTEM

The paper tape subsystem consists of the paper tape reader, handler and punch. The reader and handler is a bidirectional unit capable of reading 500 characters per second. The punch is a 150 character per second unit. The subsystem uses either paper or Mylar tape, and reads or punches 5-, 6-, 7-, and 8- channel tapes, 10 characters per inch, in widths of 11/16, 7/8, and 1 inch.

APPENDIX A
INSTRUCTION LISTING

DATA MOVEMENT			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
Load					
LDA	235	Load A	4.0	3.0	1.8
LDQ	236	Load Q	4.0	3.0	1.8
LDAQ	237	Load AQ	(M)	3.0	1.9
LDXn	22n	Load Xn	4	3.0	1.8
LREG	073	Load Registers	18.0	9.0	4.8
LCA	335	Load Complement A	4.0	3.0	1.8
LCQ	336	Load Complement Q	4.0	3.0	1.8
LCAQ	337	Load Complement AQ	(M)	3.0	1.9
LCXn	32n	Load Complement Xn	4.0	3.0	1.8
EAA	635	Effective Address to A	4.0	2.0	1.3
EAQ	636	Effective Address to Q	4.0	2.0	1.3
EAXn	62n	Effective Address to Xn	4.0	2.0	1.3
LDI	634	Load Indicator Register	4.0	3.0	1.8
Store					
STA	755	Store A	4.0	3.5	2.5
STQ	756	Store Q	4.0	3.5	2.5
STAQ	757	Store AQ	(M)	3.5	3.0
STXn	74n	Store Xn	4.0	3.5	2.5
SREG	753	Store Register	18.0	11.5	7.5
STCA	751	Store Character of A (6 Bit)	4.0	3.5	2.5
STCQ	752	Store Character of Q (6 Bit)	4.0	3.5	2.5
STBA	551	Store Character of A (9 Bit)	4.0	3.5	2.5
STBQ	552	Store Character of Q (9 Bit)	4.0	3.5	2.5
STI	754	Store Indicator Register	4.0	3.5	2.9
STT	454	Store Timer Register	4.0	3.5	2.5
SBAR	550	Store Base Address Register	4.0	3.5	2.9
STZ	450	Store Zero	4.0	3.5	2.5
STC1	554	Store Instruction Counter plus 1	4.0	3.5	2.9
STC2	750	Store Instruction Counter plus 2	4.0	3.5	2.9
Shift					
ARS	731	A Right Shift	2.5 + .2n*	2.0	1.8
QRS	732	Q Right Shift	2.5 + .2n	2.0	1.8
LRS	733	Long Right Shift	2.5 + .2n	2.0	1.8
ALS	735	A Left Shift	2.5 + .2n	2.0	1.8
QLS	736	Q Left Shift	2.5 + .2n	2.0	1.8
LLS	737	Long Left Shift	2.5 + .2n	2.0	1.8

* n = Number of shifted bit positions
(M) = Macro - operation

DATA MOVEMENT (cont)			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
Shift (cont)					
ARL	771	A Right Logic	2.5 + .2n	2.0	1.8
QRL	772	Q Right Logic	2.5 + .2n	2.0	1.8
LRL	773	Long Right Logic	2.5 + .2n	2.0	1.8
ALR	775	A Left Rotate	2.5 + .2n	2.0	1.8
QLR	776	Q Left Rotate	2.5 + .2n	2.0	1.8
LLR	777	Long Left Rotate	2.5 + .2n	2.0	1.8
FIXED-POINT ARITHMETIC					
Addition					
ADA	075	Add to A	4.0	3.0	1.8
ADQ	076	Add to Q	4.0	3.0	1.8
ADAQ	077	Add to AQ	(M)	3.0	1.9
ADXn	06n	Add to Xn	4.0	3.0	1.8
ASA	055	Add Stored to A	5.0	4.0	2.8
ASQ	056	Add Stored to Q	5.0	4.0	2.8
ASXn	04n	Add Stored to Xn	5.0	4.0	2.8
ADLA	035	Add Logic to A	4.0	3.0	1.8
ADLQ	036	Add Logic to Q	4.0	3.0	1.8
ADLAQ	037	Add Logic to AQ	(M)	3.0	1.9
ADLXn	02n	Add Logic to Xn	4.0	3.0	1.8
AWCA	071	Add with Carry to A	4.0	3.0	1.8
AWCQ	072	Add with Carry to Q	4.0	3.0	1.8
ADL	033	Add Low to AQ	4.0	3.0	1.8
AOS	054	Add One to Storage	5.0	4.0	2.8
Subtraction					
SBA	175	Subtract from A	4.0	3.0	1.8
SBQ	176	Subtract from Q	4.0	3.0	1.8
SBAQ	177	Subtract from AQ	(M)	3.0	1.9
SBXn	16n	Subtract from Xn	4.0	3.0	1.8
SSA	155	Subtract Stored from A	5.0	4.0	2.8
SSQ	156	Subtract Stored from Q	5.0	4.0	2.8
SSXn	14n	Subtract Stored from Xn	5.0	4.0	2.8

FIXED-POINT ARITHMETIC (cont)			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
Subtraction (cont)					
SBLA	135	Subtract Logic from A	4.0	3.0	1.8
SBLQ	136	Subtract Logic from Q	4.0	3.0	1.8
SBLAQ	137	Subtract Logic from AQ	(M)	3.0	1.9
SBLXn	12n	Subtract Logic from Xn	4.0	3.0	1.8
SWCA	171	Subtract with Carry from A	4.0	3.0	1.8
SWCQ	172	Subtract with Carry from Q	4.0	3.0	1.8
Multiplication					
MPY	402	Multiply Integer	11.0	7.0	7.0
MPF	401	Multiply Fraction	11.0	7.0	7.0
Division					
DIV	506	Divide Integer	19.0	14.5	14.2
DVF	507	Divide Fraction	19.0	14.5	14.2
Negate					
NEG	531	Negate A	2.0	2.0	1.3
NEGL	533	Negate Long	(M)	2.0	1.3
BOOLEAN OPERATIONS					
AND					
ANA	375	AND to A	4.0	3.0	1.8
ANQ	376	AND to Q	4.0	3.0	1.8
ANAQ	377	AND to AQ	(M)	3.0	1.9
ANXn	36n	AND to Xn	4.0	3.0	1.8
ANSA	355	AND to Storage A	5.0	4.0	2.8
ANSQ	356	AND to Storage Q	5.0	4.0	2.8
ANSXn	34n	AND to Storage Xn	5.0	4.0	2.8
OR					
ORA	275	OR to A	4.0	3.0	1.8
ORQ	276	OR to Q	4.0	3.0	1.8
ORAQ	277	OR to AQ	(M)	3.0	1.9
ORXn	26n	OR to Xn	4.0	3.0	1.8

BOOLEAN OPERATIONS (cont)			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
OR (cont)					
ORSA	255	OR to Storage A	5.0	4.0	2.8
ORSQ	256	OR to Storage Q	5.0	4.0	2.8
ORSXn	24n	OR to Storage Xn	5.0	4.0	2.8
EXCLUSIVE OR					
ERA	675	EXCLUSIVE OR to A	4.0	3.0	1.8
ERQ	676	EXCLUSIVE OR to Q	4.0	3.0	1.8
ERAQ	677	EXCLUSIVE OR to AQ	(M)	3.0	1.9
ERXn	66n	EXCLUSIVE OR to Xn	4.0	3.0	1.8
ERSA	655	EXCLUSIVE OR to Storage A	5.0	3.0	2.8
ERSQ	656	EXCLUSIVE OR to Storage Q	5.0	3.0	2.8
ERSXn	64n	EXCLUSIVE OR to Storage Xn	5.0	3.0	2.8
COMPARISON					
Compare					
CMPA	115	Compare with A	4.0	3.0	1.8
CMPQ	116	Compare with Q	4.0	3.0	1.8
CMPAQ	117	Compare with AQ	(M)	3.0	1.9
CMPXn	10n	Compare with Xn	4.0	3.0	1.8
CWL	111	Compare with Limits	4.0	3.0	2.2
CMG	405	Compare Magnitude	4.0	3.0	1.8
SZN	234	Set Zero and Negative Indicators from Memory	4.0	3.0	1.8
CMK	211	Compare Masked	4.0	3.0	2.2
Comparative AND					
CANA	315	Comparative AND with A	4.0	3.0	1.8
CANQ	316	Comparative AND with Q	4.0	3.0	1.8
CANAQ	317	Comparative AND with AQ	(M)	3.0	1.9
CANXn	30n	Comparative AND with Xn	4.0	3.0	1.8
Comparative NOT					
CNAA	215	Comparative NOT with A	4.0	3.0	1.8
CNAQ	216	Comparative NOT with Q	4.0	3.0	1.8
CNAAQ	217	Comparative NOT with AQ	(M)	3.0	1.9
CNAXn	20n	Comparative NOT with Xn	4.0	3.0	1.8

FLOATING POINT			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
Load					
FLD	431	Floating Load	(M)	3.0	1.8
DFLD	433	Double-Precision Floating Load	(M)	3.0	1.9
LDE	411	Load Exponent Register	4.0	3.0	1.8
Store					
FST	455	Floating Store	(M)	3.5	2.5
DFST	457	Double-Precision Floating Store	(M)	4.0	3.0
STE	456	Store Exponent Register	4.0	3.5	2.5
Addition					
FAD	475	Floating Add	(M)	3.0	2.7
UFA	435	Unnormalized Floating Add	(M)	3.0	2.5
DFAD	477	Double-Precision Floating Add	(M)	3.0	2.7
DUFA	437	Double-Precision Unnormalized Floating Add	(M)	3.0	2.5
ADE	415	Add to Exponent Register	4.0	3.0	1.8
Subtraction					
FSB	575	Floating Subtract	(M)	3.0	2.7
UFS	535	Unnormalized Floating Subtract	(M)	3.0	2.5
DFSB	577	Double-Precision Floating Subtract	(M)	3.0	2.7
DUFS	537	Double-Precision Unnormalized Floating Subtract	(M)	3.0	2.5
Multiplication					
FMP	461	Floating Multiply	(M)	6.0	5.9
UFM	421	Unnormalized Floating Multiply	(M)	6.0	5.7
DFMP	463	Double-Precision Floating Multiply	(M)	12.0	11.7
DUFM	423	Double-Precision Unnormalized Floating Multiply	(M)	12.0	11.5
Division					
FDV	565	Floating Divide	(M)	14.5	14.2
FDI	525	Floating Divide Inverted	(M)	14.5	14.2
DFDV	567	Double-Precision Floating Divide	(M)	23.5	23.2
DFDI	527	Double-Precision Floating Divide Inverted	(M)	23.5	23.2

FLOATING POINT (cont)			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
Negate, Normalize					
FNEG	513	Floating Negate	(M)	3.0	2.3
FNO	573	Floating Normalize	3.5 + .4n	3.0	2.3
Compare					
FCMP	515	Floating Compare	(M)	3.0	2.1
FCMG	425	Floating Compare Magnitude	(M)	3.0	2.1
DFCMP	517	Double-Precision Floating Compare	(M)	3.0	2.1
DFCMG	427	Double-Precision Floating Compare Magnitude	(M)	3.0	2.1
FSZN	430	Floating Set Zero and Negative Indicators from Memory	(M)	3.0	1.8
TRANSFER OF CONTROL					
Transfer					
TRA	710	Transfer Unconditionally	2.0	2.0	1.7
TSXn	70n	Transfer and Set Xn	2.0	3.0	1.8
TSS	715	Transfer and Set Slave Mode		2.0	1.7
RET	630	Return	4.5	4.0	3.3
Conditional Transfer					
TZE	600	Transfer on Zero	2.0	2.0	1.7
TNZ	601	Transfer on Not Zero	2.0	2.0	1.7
TMI	604	Transfer on Minus	2.0	2.0	1.7
TPL	605	Transfer on Plus	2.0	2.0	1.7
TRC	603	Transfer on Carry	2.0	2.0	1.7
TNC	602	Transfer on No Carry	2.0	2.0	1.7
TOV	617	Transfer on Overflow	2.0	2.0	1.7
TEO	614	Transfer on Exponent Overflow	2.0	2.0	1.7
TEU	615	Transfer on Exponent Underflow	2.0	2.0	1.7
TTF	607	Transfer on Tally-Runout Indicator OFF	2.0	2.0	1.7

MISCELLANEOUS OPERATIONS			M-605 Timing (μsec)	M-625, GE-625 Timing (μsec)	GE-635 Timing (μsec)
NOP	011	No Operation	4.0	2.0	1.1
DIS	616	Delay Until Interrupt Signal	2.0	2.0	1.7
BCD	505	Binary to Binary-Coded-Decimal	7.0	4.0	3.4
GTB	774	Gray to Binary	10.0	9.0	8.5
XEC	716	Execute	2.0	2.0	1.7
XED	717	Execute Double	2.0	2.0	1.7
MME	001	Master Mode Entry	2.0	3.0	2.3
DRL	022	Derail	2.0	3.0	2.3
RPT	520	Repeat	2.0	2.0	1.3
RPL	500	Repeat Link	2.0	2.0	1.3
MASTER MODE OPERATIONS					
Master Mode					
LBAR	230	Load Base Address Register	4.0	3.0	1.8
LDT	637	Load Timer Register	4.0	3.0	1.8
SMIC	451	Set Memory Controller Interrupt Cells	5.0	3.0	1.8
Master Mode and Control Processor					
RMCM	233	Read Memory Controller Mask Registers	4.0	3.0	1.9
RMFP	633	Read Memory File Protect Register	4.0	3.0	1.9
SMCM	553	Set Memory Controller Mask Registers	5.0	3.0	1.8
SMFP	453	Set Memory File Protect Register	5.0	3.0	1.8
CIOC	015	Connect I/O Channel	4.0	3.0	1.8

APPENDIX B

STANDARD GENERAL ELECTRIC CHARACTER SET

Standard Character Set	GE-Internal Machine Code	Octal Code	Hollerith Card Code	Standard Character Set	GE-Internal Machine Code	Octal Code	Hollerith Card Code
0	00 0000	00	0	↑	10 0000	40	11-0
1	00 0001	01	1	J	10 0001	41	11-1
2	00 0010	02	2	K	10 0010	42	11-2
3	00 0011	03	3	L	10 0011	43	11-3
4	00 0100	04	4	M	10 0100	44	11-4
5	00 0101	05	5	N	10 0101	45	11-5
6	00 0110	06	6	O	10 0110	46	11-6
7	00 0111	07	7	P	10 0111	47	11-7
8	00 1000	10	8	Q	10 1000	50	11-8
9	00 1001	11	9	R	10 1001	51	11-9
[	00 1010	12	2-8	-	10 1010	52	11
#	00 1011	13	3-8	$	10 1011	53	11-3-8
@	00 1100	14	4-8	*	10 1100	54	11-4-8
:	00 1101	15	5-8	)	10 1101	55	11-5-8
>	00 1110	16	6-8	;	10 1110	56	11-6-8
?	00 1111	17	7-8	'	10 1111	57	11-7-8
↻	01 0000	20	(blank)	+	11 0000	60	12-0
A	01 0001	21	12-1	/	11 0001	61	0-1
B	01 0010	22	12-2	S	11 0010	62	0-2
C	01 0011	23	12-3	T	11 0011	63	0-3
D	01 0100	24	12-4	U	11 0100	64	0-4
E	01 0101	25	12-5	V	11 0101	65	0-5
F	01 0110	26	12-6	W	11 0110	66	0-6
G	01 0111	27	12-7	X	11 0111	67	0-7
H	01 1000	30	12-8	Y	11 1000	70	0-8
I	01 1001	31	12-9	Z	11 1001	71	0-9
&	01 1010	32	12	←	11 1010	72	0-2-8
.	01 1011	33	12-3-8	,	11 1011	73	0-3-8
]	01 1100	34	12-4-8	%	11 1100	74	0-4-8
(	01 1101	35	12-5-8	=	11 1101	75	0-5-8
<	01 1110	36	12-6-8	"	11 1110	76	0-6-8
\	01 1111	37	12-7-8	!	11 1111	77	0-7-8

Progress Is Our Most Important Product

GENERAL ELECTRIC

RADIO GUIDANCE OPERATION • SYRACUSE, N. Y.

RGO-156/6-65