GE-645
System Manual

GENERAL ELECTRIC

CPB-1231

GE-645
SYSTEM MANUAL

April 1966

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

PREFACE

Part I of this manual provides an introduction to the GE-645 Information Processing System. The first three chapters analyse the thinking which brought about the development of an information processing utility. The GE-645 makes computer service available in the same way electricity is made available simultaneously to many users. Chapter 4 provides a summary which allows the readers to obtain a general understanding of how the utility concept is implemented.

Part II provides moderately detailed information about the major hardware and software components of the system.

Appendix A describes the ASCII code which is used with the GE-645. Appendix B contains an alphabetized summary of acronyms. A subject index is provided to permit the manual to be used for reference purposes.

Comments on this publication may be addressed to Engineering Publications Standards, Computer Equipment Department, General Electric Company, 13430 North Black Canyon Highway, Phoenix, Arizona, 85029.

ACKNOWLEDGMENT

MULTICS (Multiplex Information and Computing Service) is an operating program developed from research by Massachusetts Institute of Technology, Bell Telephone Laboratories, and General Electric. It draws upon the design and operating experience gained with CTSS (Compatible Time-Sharing System) at the Massachusetts Institute of Technology Computation Center and Project MAC. The commercial product being derived by the General Electric Company from this effort is known as Multics-645/I which is an operating system for the GE-645 computer.

CONTENTS

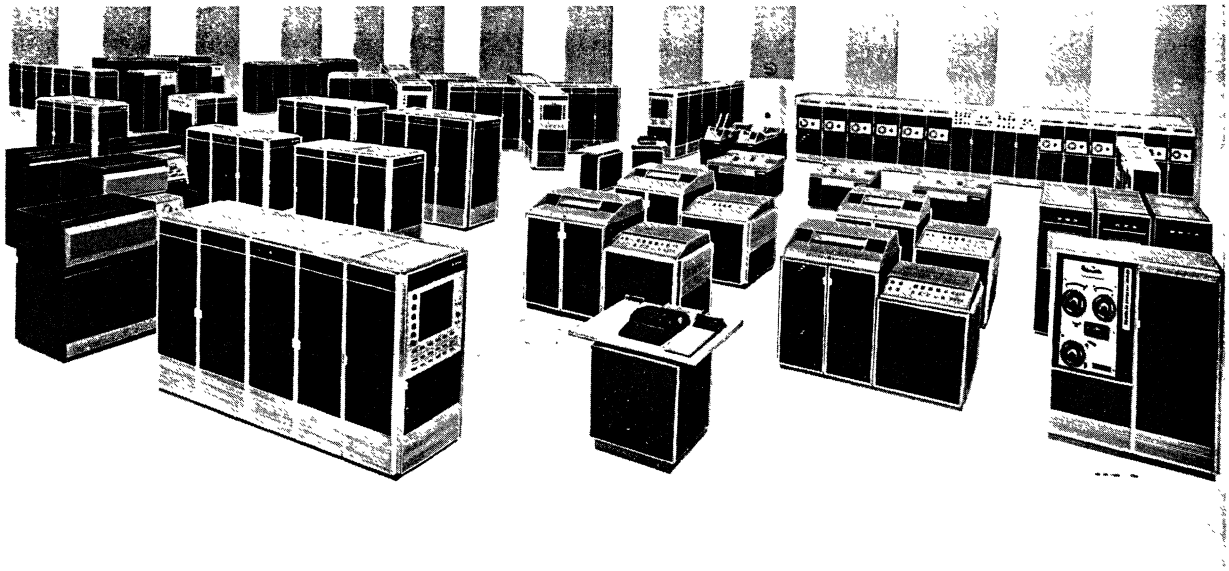
PART I. Introduction to the GE-645 System

PART II. GE-645 System Description

Page

ILLUSTRATIONS

PART I.
Introduction to the GE-645 System



GE-645 SYSTEM

1. PROBLEM STATEMENT

During the early growth period of computers, a computer was a unique and complex piece of equipment, but was too expensive to give to each person with a problem. In order to justify the cost of a computer and to take advantage of its speed, batch processing became the standard mode of operation. A staff of experts was required to manage, program, and operate the computer. As a result, the man with the problem gradually had barriers placed before him, and he became more and more removed from the computer.

The computer language was a primary barrier. The man with the problem didn't know how to state the problem in a language acceptable to the computer. Specialists were called in who understood the computer languages, but didn't understand the problem. As the problem passed from specialist to specialist, it went through several stages of translation. The elapsed time from problem statement to problem solution was lengthened, and often a clear statement of the original problem was lost in the several translations.

The layers of computer specialists became thicker, and the computer gradually became less accessible to the man with a problem. There was a great deal of concentration on the computer itself, and much less on the methods for solving individual problems. Many times, the person with a problem must have said, "I'd put more work on the computer if it weren't such a chore." In frustration, he used methods not well suited to the needs of the problem.

Computer technology evolved at a rapid pace. However, most of the knowledge was concerned with computer equipment and computer techniques, and very little was concerned with the techniques required to make the computer a convenient problem-solving tool.

As middle and upper management took a broader view of the problems, the need to share programs and data files among several groups became apparent. In some instances, different groups worked on separate portions of the same large problem and each group developed its own set of programs and data files. In other situations, hardware and software limitations made it difficult for one group to utilize the files of another.

A tremendous amount of this information should have been centralized and made available to several different groups at the same time. This lack of integration led to wasteful duplication of effort and fell short in supplying the total information overview sought by management.

The computer schedule was another barrier placed between the computer and the man with a problem. A computer was scheduled in order to get maximum use of the equipment. However, the entire computer system became unavailable when certain peripherals (e.g., typewriter or card reader) became unavailable. At other times the computer system was idle for many minutes waiting for tapes to be mounted or cards to be placed in the card reader. Regardless of priorities, all jobs were put aside for demonstrations, maintenance, and real-time problems.

The tight schedules set for maximum use of equipment were not realistic, were difficult to enforce, and caused almost everyone to endure a long turnaround time waiting for the results.

The allocation of computer time and storage media was another problem closely associated with scheduling. Did management really have control of the costs in the computer system? A junior programmer could easily accumulate 50 reels of magnetic tape, collect drawer after drawer of cards, and use $5,000 worth of computer time. Who determined when it was more economical to store data in magnetic core, on magnetic tape, or on magnetic drum? Decisions were based on an individual's past experience rather than on statistics gathered from group experience over a longer period of time.

New and expanding computer applications required users to increase the size of their hardware systems. However, this was difficult and expensive when whole systems had to be added or replaced. Without hardware modularity, an entire hardware system consisting of all three capabilities (processing, storage, input/output) had to be added to provide more of any one capability. To take advantage of the new hardware configuration, applications had to be reprogrammed, often at considerable cost to the user.

When performance characteristics of separate computer components were examined and analyzed in conjunction with the requirements of the problems being solved, it was found that some parts were idle while other parts operated at full speed. During calculations for a scientific problem, the input/output section worked only intermittently while the processor worked continuously; however, during calculations for a business problem, the processor worked part time while the input/output was kept busy.

The evolution and growth of the computer has not stopped. The technology associated with computers and their application continues to evolve at an ever increasing rate. In order for the computer to continue to serve the needs of business, industry, science, education, and government; the barriers between the computer and the man with a problem must be eliminated.

2. AN INFORMATION PROCESSING UTILITY

An information processing utility can eliminate many of the factors that have separated the computer from the man with a problem. Such a utility is based on a new mode of thinking in the use of computers. Instead of looking upon the computer as an expensive calculator for highly trained specialists, one can look on it as a facility which can provide service to any man with a problem. The service rendered is information processing in the broadest sense. The utility center plays a role analagous to that of a power station providing electricity to a community. The computer is the source of computing power. The design of the information processing utility permits distribution of information processing services to users when required, where required, and in an economical, reliable, and easy-to-use manner.

Much of the necessary technology already exists. Many types of input/output terminals can be connected to the computer. Systems exist in which several users can have access to a computer and to its resources from different locations at the same time. Remote terminals (e.g. unit record devices, teletypewriters, graphic displays) can be placed at locations convenient for the user. The problem is to integrate existing experimental approaches into a complete, reliable, and economical system.

An information processing utility must provide:

- Dependable operation
- Remote terminals to communicate with the utility center
- Simple user-oriented languages for communication of problems and solutions
- Rapid turnaround time to permit direct interaction with the computer
- Full batch processing capability
- Concurrent operation of interactive jobs and batch processing jobs with high efficiency
- Memory management that frees the user from any concern about how to fit his program and data into the system
- Input/output management that frees the user from any concern about input/output operations, unless he wishes otherwise
- A storage system which permits easy sharing of data and programs
- File protection mechanisms to ensure that access to files is restricted as specified by the owners
- Modular hardware to meet the user's changing needs

- Modular software to permit its dynamic modification
- Mechanisms to permit management control over resource usage
- Bookkeeping procedures to record resource usage and charge users appropriately
- Clear, readable reference and training material to guide users, operators, maintenance engineers, and adminstrators in the use and control of the system.

3. GE-645 DESIGN CONCEPTS

New system designs to meet the objectives of an information processing utility were needed for both hardware and software. The design of separate hardware and software features had to be carefully integrated to produce all of the desired capabilities in a single system.

The GE-645 Information Processing System with its operating system (Multics-645/I) is General Electric's design of an information processing utility.

With a GE-645, the man with a problem can often do his own programming. He can solve his problem using a terminal conveniently located in his home or office. After learning a few simple rules and some phrases called "commands," he can guide the computer through the steps of his problem solution from the keyboard of his remote terminal. If he is a novice in programming, he can obtain a useful knowledge of one of the simpler programming languages in a few hours. If he is an experienced programmer and wants greater power and flexibility, he can employ any of the widely used languages in the industry.

The full capability of the information processing utility is as near to the man with a problem as the nearest telephone outlet. The tasks which he requests to be performed will normally take very little computer time to execute. Thus, each user can proceed in the solution of his problem without being delayed by the computer. He is allowed to "interact" with the computer as it achieves the results he desires. The vast resources of the system are his to command. While using these resources, he has the impression that he is the sole user.

Even when the GE-645, operating under Multics-645/I, is interacting with users at remote terminals, it can concurrently serve users with batch processing work. The system can also guarantee responses within rigidly fixed time intervals. This feature makes the system suitable for execution of real-time processes.

When operating under control of Multics-645/I, a user is not concerned about the size of his program or data storage. Multics-645/I will automatically move unused parts of his program to secondary storage. Hence, both the information currently in core memory and that which Multics-645/I has placed in secondary storage are available to the user. In spite of the actual physical location of user programs and data, they appear to him as if they were all in core memory.

Multics-645/I allows a user to declare the degree of privacy to be observed in the use of each of his files. He may declare a file to be private, public, or accessible only to a specific list of

other users. He can further restrict the use of a file by declaring the manner in which it may be accessed. For example, a file containing a program could be made public for purposes of execution but private for purposes of reading and modification.

A user may find it convenient to use subprograms or data which are provided by Multics-645/I or made available to him by other users. In fact, some programs and data files may be needed concurrently by many users. Multics-645/I enables many users to share programs and data files. In essence, the system provides storage facilities analogous to the combined resources of a public library (public system files), the library of co-workers (shared files), and a personal library (private files).

In the GE-645, reference material and documents to guide the user are an important part of the overall system. In addition to the usual technical documents and programming language manuals, specific documents are included to provide the necessary background information to a selected audience, such as tutorial documents and suggested reading lists for trainees, administrators, operators, and programmers. Some of the documents are being collected in computer accessible form in the GE-645 storage hierarchy.

The many powerful and convenient features of the GE-645 enable the man with a problem to obtain results faster and more effectively than has previously been possible. The use of this system should result in significant overall economy. The cost to the user is determined in part by the number of users who can take advantage of the service of the information processing utility. The special features of the GE-645 allow the utility to be used by many subscribers without sacrificing the performance offered to them individually.

Multiprogramming allows adequate service to be provided to all classes of users. It permits processing part of one program, then part of a second, then part of a third, etc. This allows multiple programs to receive service in rapid succession giving the appearance that they are all being executed simultaneously. If this technique were not used, each user would need to await the completion of many other user programs before having an opportunity to utilize a processor. Without multiprogramming, interactive users would experience intolerable delays.

The total processing capability of the system is increased by having multiple processors execute multiple programs concurrently. This technique is known as multiprocessing.

The control of the processors is directed from one user program to another by Multics-645/I. Every time a new program is placed in execution, special registers are initialized which prevent damage to information belonging to any other user or the operating system.

Large memories are made available in the GE-645 hardware. Through their use, many programs and their data may be stored concurrently awaiting execution. Even with the large memories available on the GE-645, it is generally impossible to allow the programs and data of all users to reside in core memory throughout their execution. The number of users that can be serviced by the GE-645 is made large by providing a high-speed drum for moving information rapidly into and out of core memory. The number of users that can be serviced is also increased by employing the techniques of paging and segmentation.

In the concept of paging, a program is divided into pieces of equal size called pages. When the program is in execution, only its most active pages reside in core storage. Its less active pages are stored in the secondary storage hierarchy.

In the GE-645 the management of paged programs is facilitated by special hardware which makes the pages appear to be in contiguous locations even though they may be widely separated in core memory. The hardware also keeps track of which pages have been altered while they are in core memory so that unaltered pages need not be rerecorded in secondary storage when they are removed from core memory.

Segmentation allows a program to be divided into separate parts called segments. Each segment can be thought of as a separate core memory with its own origin and maximum size. One of the important features of segments is that they may be shared among many users. Hence, only a single copy of a program or data file need exist if these are properly stored in the segment being accessed by many users.

A second important property of segments is that they allow a user to collect in effectively separate memories information with like characteristics. For example, some or all of the unalterable information of a program can be brought together into a "read only" segment. This not only protects the information from inadvertent modification but also collects information which will never need to be rewritten to secondary storage during the paging operation.

Modular hardware lets the user reconfigure a system to fit the changing needs brought about by growth. A system can consist of multiple memory, processor, and input/output controller modules. The modular hardware lets the user assemble a combination of modules that most closely fits his current need. This reconfiguration can take place without providing an excess of unwanted modules. For example, one processor module can be added when additional processing is needed, or one input/output controller and/or peripheral devices can be added when greater input/output capability is needed.

Modular software permits dynamic modification of the operating system. Software modularity permits a utility center administrator to alter the operating system to fit his computing needs. It also allows the operating system maintenance group to make appropriate improvements and modifications that are of benefit to the user. The same programming standards are used in Multics-645/I as in user programs so that maximum consistency and flexibility is achieved within the system.

4. SYSTEM SUMMARY

The previous chapters have discussed the general problems facing the user, the objectives of the GE-645 system, and the design concepts which enable the system to meet its objectives and solve the user's problems. This chapter presents a summary of the parts of the system. Should the reader desire more information than is contained in this chapter, he may refer to Part II.

This chapter contains a brief view of the total hardware-software system followed by descriptions of the hardware and its components, the software system and its components, and the documentation for the hardware and software.

TOTAL SYSTEM

The GE-645 is a large-scale, high-performance, binary computing system that operates in fully-integrated harmony with the powerful Multics-645/I operating system, several language processors, and a wide variety of application packages. Most of the equipment is located at the utility center. Terminals are placed in locations convenient for the users of the system.

HARDWARE

The hardware consists of four types of modules, peripheral and terminal equipment, and common carrier facilities for the terminals. The four module types are processor, generalized input/output controller (GIOC), memory, and drum. A full range of peripherals is available, from perforated tape units to large, high-speed discs. Terminals that may be used range from slow speed teletypewriters to complex display consoles and computers. Terminals are connected to the system with standard, presently-available communication equipment.

A typical hardware system organization is shown in Figure 1. In this example, two processors, four memories, two GIOCs, and one drum are combined to form a nine-module system. The figure shows the interconnections between the nine modules and the way that the peripherals and terminals are connected to the system.

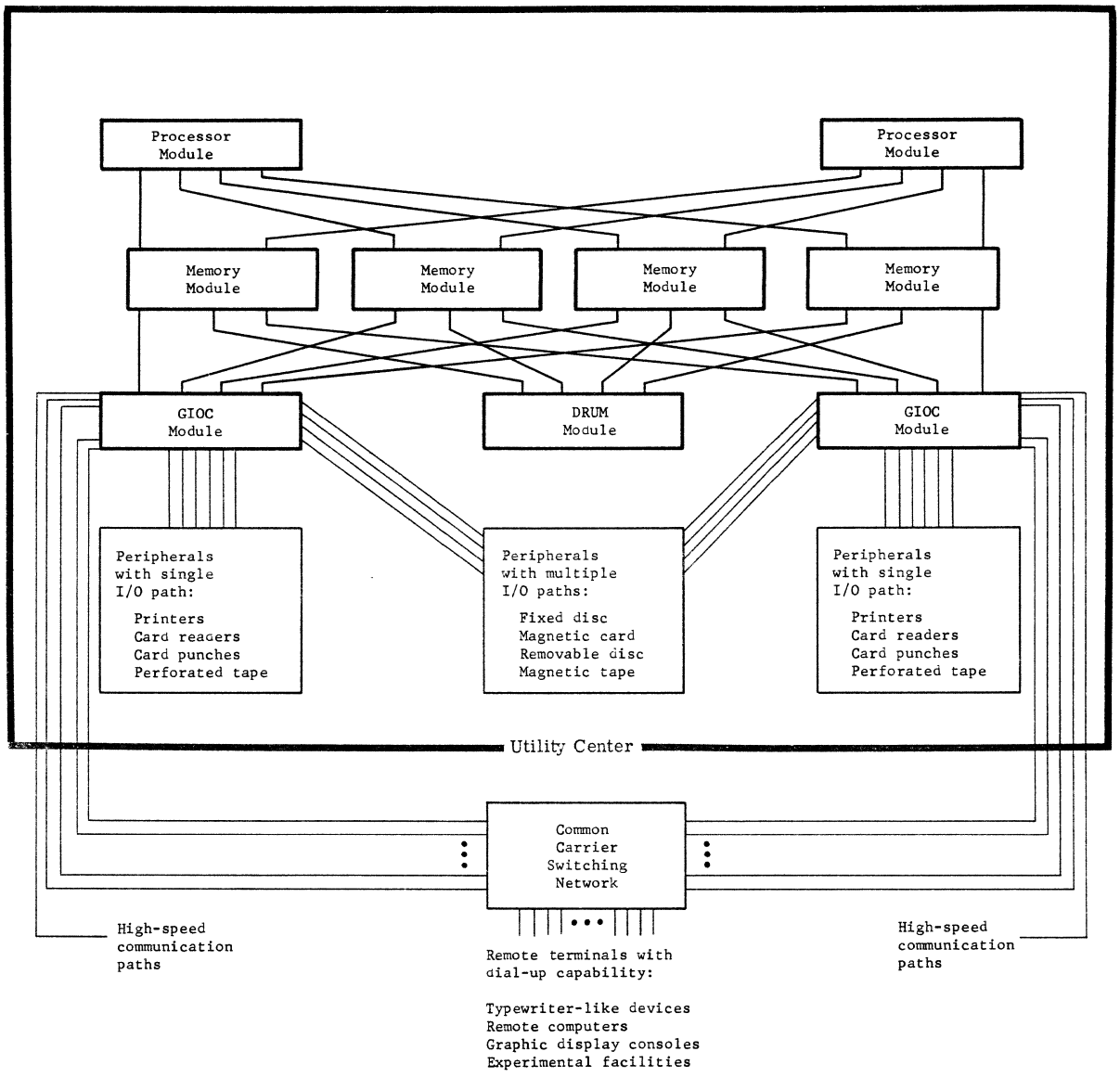


Figure 1. Hardware System Organization

The following paragraphs discuss the principal characteristics of the major units shown in Figure 1.

The system control console continuously displays the status of each module and peripheral subsystem. A teletypewriter provides a printed copy of the "conversations" between the operator and the operating system.

The processor modules do the computation and decision-making in the system. Many registers and an associative (content-addressable) memory are provided in each processor to facilitate rapid execution of powerful instructions. Performance is also enhanced with instruction overlap: that is, address preparation for the next instruction and fetching of subsequent instructions continue while the current instruction is being executed. The instruction repertoire is large and varied, and has 402 operation codes currently defined. (This leaves 110 operation codes reserved for future upward growth.) A wide range of conventional instructions is supplemented with many special-purpose instructions and several instructions specifically for character manipulation. Address modification includes conventional register modification and multilevel indirect addressing plus several variations particularly aimed at character manipulation and operating on stacks. All hardware manipulation related to segmentation and paging is done by a processor.

The GIOC modules handle all input/output for peripherals and terminals. (Note that the drum module is independent of the GIOC.) Input/output operations on a given GIOC are performed concurrently but independently of each other. The detailed actions to be performed during an input/output operation are specified in a list of control words stored in memory. Once a processor sets up the control words and initiates an input/output operation, the GIOC obtains the control words from memory and transfers data without any further processor action.

Memory modules contain the system's high-speed storage, the system's time clocks, and serve as control communication paths between the processor, GIOC, and drum. A memory module consists of 32k, 64k, or 128k 36-bit words with a storage cycle time of one microsecond. The system can move data in groups of 6, 9, 36, or 72 bits into or out of memory. There are two types of clocks, a calendar/read time clock and an alarm clock. The clocks are located in a memory module so that all processors can use the same clocks. Thus, all processors use the same time base and perform time calculations in a consistent manner. Each memory module also contains a group of "interrupt cells." These cells are used by a processor, GIOC, and drum module to notify the operating system of the occurrence of special events.

Memory is addressed in an interlaced manner. When a block of data is transferred in or out of memory, the data accesses are distributed among the memory modules. This prevents a data transfer to the drum, for example, from putting a heavy load on just one memory module and thereby effectively preventing other modules from using that memory module.

A drum module is used as an extension of memory. Drum capacities range from one to sixteen million words, with four million words being a typical size. The drum performance characteristics were carefully designed to match the drum to the GE-645 system. The block sizes used for information transfer are the same size as the pages handled by the processor: 64 or 1024 words. A transfer rate of 470,000 words per second was chosen to minimize the time required to move data between drum and memory, yet not be so fast that memory is overloaded when a drum is transferring data.

The kinds of peripheral and remote terminal equipment associated with the GE-645 are:

Removable disc
Large, fixed disc
Magnetic card
Magnetic tape
Card reader
Card punch
Printer
Perforated tape
Typewriter-like devices (both hard copy and cathode ray tube display)
Other computers (for both normal I/O and for experimental facilities.)
Graphic displays

The system shown in Figure 1 has nine modules. The system can be expanded to a maximum of sixteen modules: eight memory modules and a combination of eight processor, GIOC, and drum modules.

The separation of the major system functions into several modules provides hardware redundancy in case of equipment failure and facilitates system growth. The proper quantities of the various types of modules may be combined to create a system that is well suited to the workload of each utility center. Peripherals and terminals also have this modular property. Peripherals which operate at high speeds and require little operator attention are connected to both GIOCs to improve system performance and provide more than one path in case a GIOC should ever malfunction.

SOFTWARE

The GE-645 software consists of the Multics-645/I operating system, language processors, application packages, and compatible GE-625/635 series programs. The general capabilities of these will be described briefly.

Operating System

MULTICS (Multiplex Information and Computing Service) is an operating program developed from research by Massachusetts Institute of Technology, Bell Telephone Laboratories, and General Electric. It draws upon the design and operating experience gained with CTSS (Compatible Time-Sharing System) at the Massachusetts Institute of Technology Computation Center and Project MAC. The commercial product being derived by the General Electric Company from this effort is known as Multics-645/I which is an operating system for the GE-645 computer.

The four major elements of Multics-645/I are the supervisor, the command system, the file system, and the input/output system. These major elements are briefly described below.

The supervisor determines the sequence in which various user programs and the Multics-645/I modules serving them are executed. Each user is alloted a fair share of available processor time by the supervisor. The supervisor contains the scheduler. Under normal circumstances the scheduler guarantees that interactive users may proceed at full speed and that batch processing jobs are completed before the specified deadlines. The supervisor also measures and records the amount of system resources expended by each user.

The command system examines the input from a user's terminal looking for a command and its arguments, such as "typeout mydata." When a command and its arguments are found, they are changed from user-oriented format into hardware-executable format. The module that executes the command is then called. The command system can be used by batch users by substituting a file containing commands for a terminal.

The file system frees the user from concern over the physical location of any of his information. He refers to his programs or data by name. The file system stores and catalogs them in its storage hierarchy. If the program or data is not in core memory when it is referenced, it is automatically retrieved and made available. The file system allows all users to retain as much information as they wish in machine-accessible secondary storage. In most cases the use of removable media such as cards and magnetic tape is unnecessary. It moves little-used information to devices with longer access time to allow ample space on faster devices for more frequently-used files. The file system provides information to a user when he requests it and protects it from accidental destruction. At the same time the file system allows files to be shared among authorized users.

The input/output system uses the GIOC to perform all reading and writing of information to peripheral and terminal devices. It provides a way for users to communicate with specific devices in cases where the device-independent input/output of the file system is not suitable. The input/output system performs the code conversion, queuing, and buffering needed by specific devices.

Language Processors

Language processors for all the well known and widely used languages in the industry are provided with the GE-645. This makes the system immediately useful for almost anyone. Even the needs of the novice programmer are provided for by BASIC (developed by Dartmouth College). The language processors are:

BASIC (the Beginners All-purpose Symbolic Instruction Code)
FORTRAN IV
PL/I
GE-645 Assembler
ALGOL
COBOL
JOVIAL
GIFT (FORTRAN II to FORTRAN IV)

Application Packages

A rich heritage of time-tested programs is brought forward from the GE-635 to the GE-645. Many of these are application packages such as:

Linear Programming
Automatically Programmed Tools (APT)
APT Postprocessors:
- Monarch and Jones and Lamson
- Burgmaster
- Milwaukee-Matic (Models II and III)
- Bullard
- Giddings and Lewis

Math Routines
SIMSCRIPT
Media Conversion
BMD Statistical Programs
PERT/TIME
PERT/COST
Integrated Data Store (IDS)
Sort/Merge

GE-625/635 Compatibility

Compatibility with GE-625/635 programs is provided at the object program level. This compatibility is achieved partially through the similarity of instructions between the members of the GE-600 line. In addition, interfaces are provided within Multics-645/I which are identical to those provided to users at execution time when operating under GECOS II, the standard operating system for the GE-625/635.

DOCUMENTATION

The GE-645 documentation includes the customary hardware reference manuals, software reference manuals, and operational manuals. Tutorial documents and technical papers are also included. A comprehensive reading guide assists the reader in selecting documents at a level of detail suited to his immediate needs. Some of the documents are prepared and maintained on a GE-645 system and, therefore, will be accessible from a remote terminal.

PART II.
GE-645 System Description

5. HARDWARE SYSTEM CHARACTERISTICS

This chapter discusses several overall system features and implications so that the reader may better understand the total hardware system. The topics covered in this chapter include hardware system organization, equipment suppliers, various system configurations, and connecting input/output devices to the system.

HARDWARE SYSTEM ORGANIZATION

Figure 2 is identical to Figure 1 of Chapter 4 and is reproduced for the convenience of the reader. The typical system is described here with much more detail than it was in Chapter 4.

Considerable modularity is provided in the system for several reasons. The user may select various modules and peripherals in order to tailor his system to his particular work load. Modules and peripherals may be added to a system at any time without changing the users' programs or the software. Only configuration tables in Multics-645/I need to be changed. The user system may also grow in a different way. An existing module or peripheral may be replaced by a higher performance module or peripheral of the same type, yielding improved system performance. Considerable hardware redundancy also exists in a typical system so that if, for example, a processor module should malfunction and have to be removed from the system, a processor module is still left in the system so that the system can continue operation.

The system in Figure 2 has multiple paths to many of the modules, peripherals, and terminals. Multiple paths between modules increase system performance by permitting memory interlacing. (See Chapter 8, Memory Module, for details.) Multiple paths to peripherals increase system performance. For example, both GIOCs can perform input/output operations with a disc simultaneously. Another significant advantage of multiple paths becomes evident when one considers hardware malfunctions. Most components of the system have more than one path to them; therefore, the malfunctioning of one component does not isolate any multiple-path component.

Peripherals such as printers and card readers are ordinarily connected to only one GIOC because they are allocated to only one user at a time. An operator can, for example, move an operation from a printer on one GIOC to a printer on a different GIOC if the first printer or its GIOC should have a problem. If desired, a single input/output path peripheral can be connected to both GIOCs. This requires another hardware unit, a peripheral switch, which is discussed later in this chapter.

The paths between modules, peripherals, and terminals pass through "ports" in modules and "channels" in GIOCs. Ports and channels are terms for the hardware at the boundary of a module and are illustrated in Figure 3.

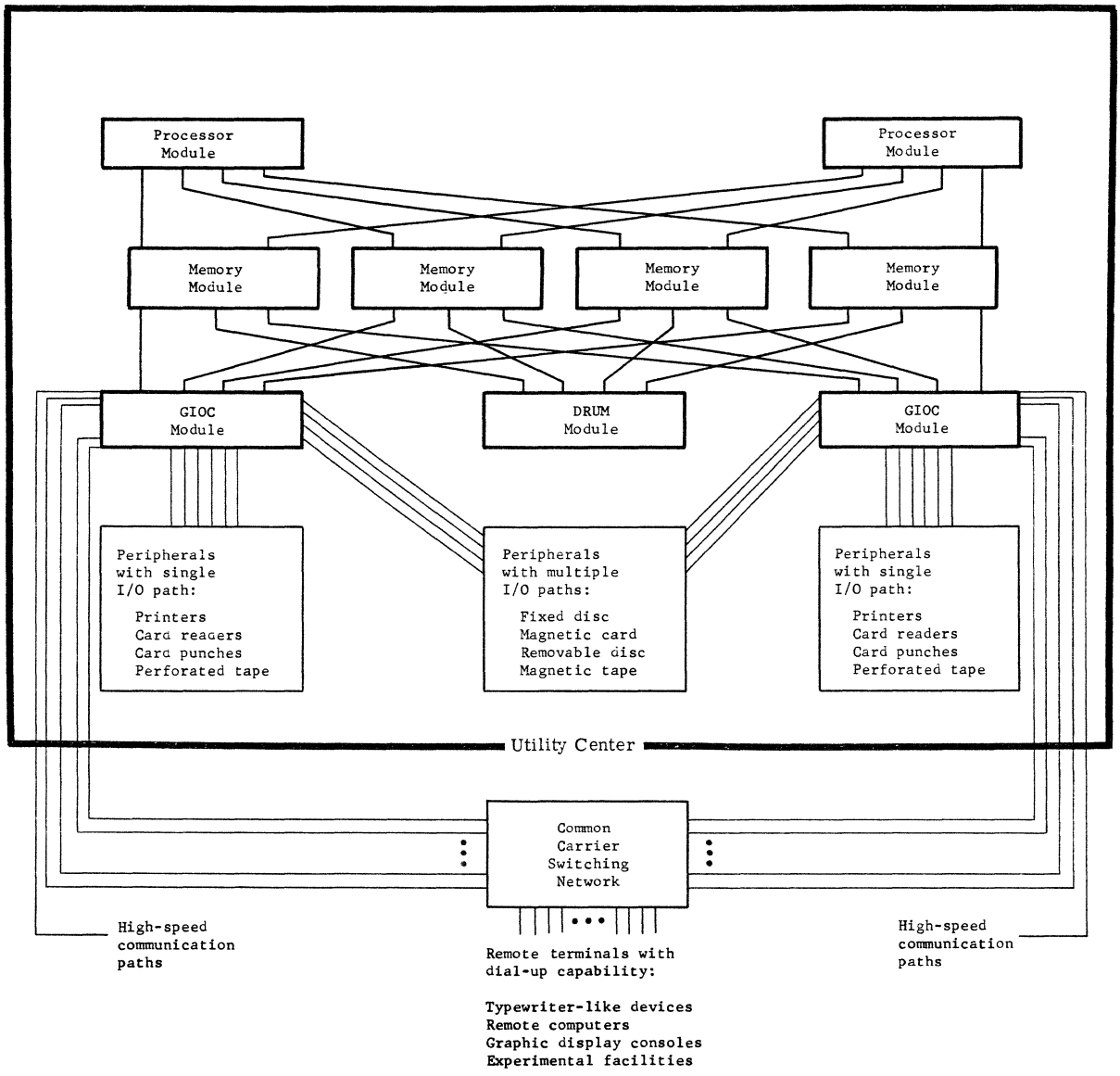


Figure 2. Hardware System Organization

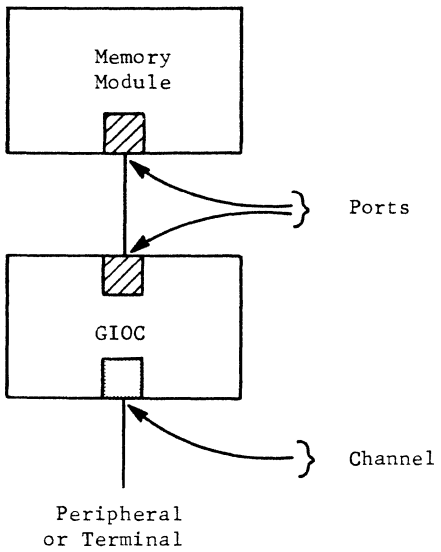


Figure 3. Distinction Between Ports and Channels.

All types of modules--processor, GIOC, memory, and drum--can have up to eight ports. Thus, the maximum number of modules in a system is sixteen: eight memory modules, and eight modules of some combination of processors, GIOCs, and drums.

Control of the system involves the system control console and one or more teletypewriters. The system control console displays a summary of the status of every module and every peripheral subsystem at the utility center. Portions of the display will be of interest to the operator. The entire display will be of interest to the field engineer should a major malfunction occur. Another major function of the system control console is the provision of a way to initialize the system and start the bootloading operation from one central point.

Although a teletypewriter is not physically part of the system control console, it is a necessary part of the total system control. More than one teletypewriter may be used, with messages for certain units sent to a teletypewriter that is physically located near those units. Thus, printer messages would be typed near the printers and messages for overall control of the system would be typed near the system control console. The printed record of the conversations between the operator and the operating system can be quite valuable in monitoring system performance and in troubleshooting certain malfunctions. Each of the teletypewriters is connected to a GIOC through a standard communication line interface.

Processors, GIOCs, and drums all have 24 bits for the formation of absolute addresses. This permits use of over 16 million words of memory. It is not anticipated that memories of this size will be available for initial GE-645 systems. However, the 16-million-word addressing capability does exist to facilitate growth of the system.

The General Electric Company supplies almost all of the equipment at the utility center. The heavy line near the bottom of Figure 2 helps to define the boundary of the utility center. All of the equipment above the line is physically located at the central computer installation, and comprises what is normally thought of as the utility center equipment. There will be a few terminals that are connected into the common carrier switching network but are actually physically located with the utility center equipment.

Table 1, Equipment Suppliers, defines which company provides or is responsible for the various major parts of a system. The abbreviations used in the table to identify the company are:

GE	Computer Equipment Department, General Electric Co.
Customer	The company that leases or purchases the utility center equipment from General Electric.

User — The company that uses the services provided by the utility center. The user and customer companies may be the same. Or, some users may be part of the customer company and the remaining users may be part of one or several other companies.

TABLE 1
EQUIPMENT SUPPLIERS

Item	Provided By
System Control Console	GE
Modules (processor, GIOC, memory, drum)	GE
Connections between modules	GE
Peripherals	GE
Connections between GIOC and peripherals	GE
Terminals	GE/Customer/User
Data sets at terminals	Customer/User
Communication lines for terminals	Customer/User
Common carrier switching network	Customer
Connections between common carrier switching network and data sets at utility center	Customer
Data sets at utility center	Customer
Connections between data sets at utility center and GIOC	Customer
Physical facilities at utility center	Customer
Electrical power at utility center	Customer
Motor-generator set at utility center	GE

SYSTEM CONFIGURATIONS

Most of the discussions so far in this manual have centered around Figures 1 and 2. Although this figure represents a typical system, there are obviously many other possible system configurations. Some of these are shown in Table 2, on the following page, where the middle column indicates the quantities that might be present in the typical system of Figure 2. Since the number of terminals that may be connected to the system is not directly reflected in the amount of hardware present in the utility center, what is shown in the table for terminals is the number of terminal-type input/output channels in the total system.

It is quite important that the reader realize that the quantities shown in Table 2 illustrate only three of the many possible system configurations. The actual quantities to order for a particular

installation depend strongly on the workload and applications to be run at that installation. One installation may need more mass storage and fewer terminal channels than the quantities shown, while another installation might need more printers and magnetic tapes and fewer terminal channels.

TABLE 2
SYSTEM CONFIGURATIONS

	Quantities		
	Small	Typical (Fig. 2)	Large (not maximum)
System Control Console	1	1	1
Processor	1	2	4
GIOC	1	2	3
Memory			
Number of modules	2	4	8
Total memory	128k	256k	1M
Drum			
Number of modules	1	1	1
Total capacity (words)	2M	4M	16M
Fixed disc (words)	33M	67M	133M
Removable disc (words)	-	10M	20M
Magnetic card (words)	-	113M	454M
Magnetic tape handlers	4	16	32
Printers	2	4	6
Card Readers	1	2	3
Card Punches	1	2	2
Perforated Tape	-	1	2
Channels for low speed teletypewriters	64	192	384
Channels for voice-grade communication lines for remote terminals such as DATANET*-760 GE-115	-	12	18

k = thousand
M = million

*Reg. Trademark of General Electric Company

CONNECTING INPUT/OUTPUT DEVICES TO THE SYSTEM

There are several ways to connect both peripherals and terminals to a GE-645 system. These are briefly discussed to show the reader how peripherals and terminals may be connected to best suit his needs.

Figure 4 illustrates the connection of single-channel peripherals to a GIOC. Both single device (card reader) and multiple device (magnetic tape) subsystems are shown.

System performance is improved by using more than one input/output channel for multiple-device subsystems. An example is the dual channel magnetic tape subsystem shown in Figure 5. Two independent operations may take place simultaneously, and either GIOC may use any tape handler.

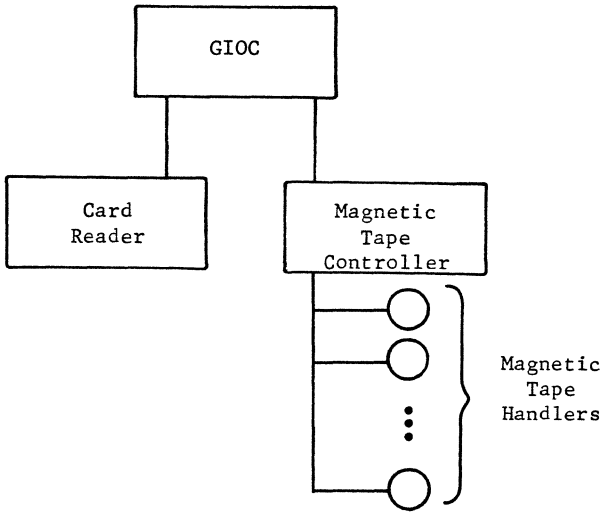


Figure 4. Single-Channel Peripherals Connected to a GIOC

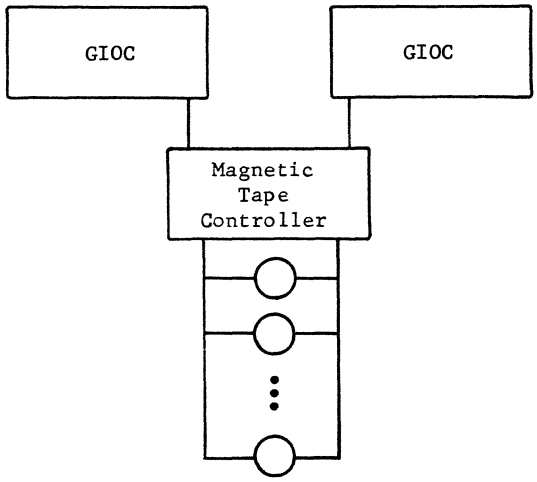


Figure 5. Multiple-Channel Peripheral Connected to two GIOC's

Additional flexibility is available by using a peripheral switch. Details on peripheral switches are given in Chapter 10. Briefly, a peripheral switch provides a rapid way to disconnect a peripheral from one GIOC and connect it to another GIOC. Two examples of this are given in Figures 6 and 7.

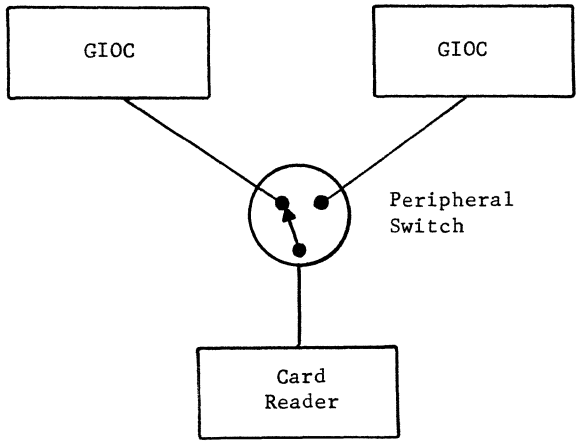


Figure 6. A Peripheral Switch and Two GIOC'S

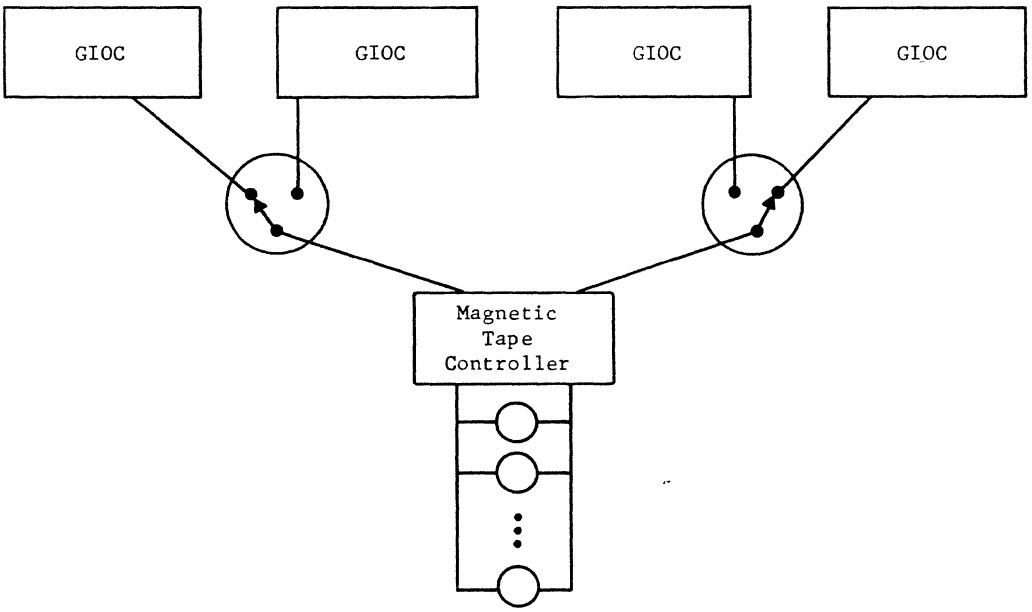


Figure 7. Two Peripheral Switches and Four GIOC's

Terminals may also be connected to the system in several ways. In general, the user at a terminal will dial into the system in order to become connected. There are many possible paths through the switching network. Depending on the path taken, the user may be connected to either GIOC. And since many of the paths go to the same GIOC, his connection may be to any one of several channels within the GIOC. In order to dial in, the user must have some kind of a data set in his terminal. (This is the usual situation.) The general data set usage is illustrated in Figure 8.

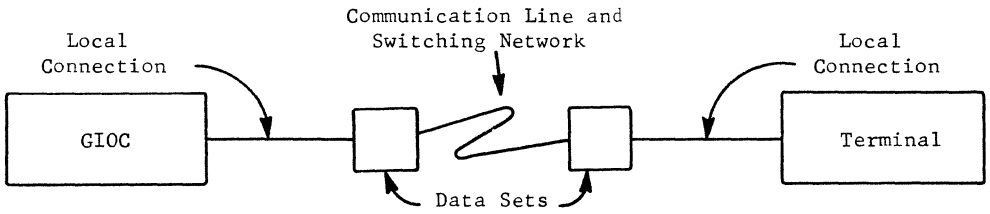


Figure 8. General Data Set Usage

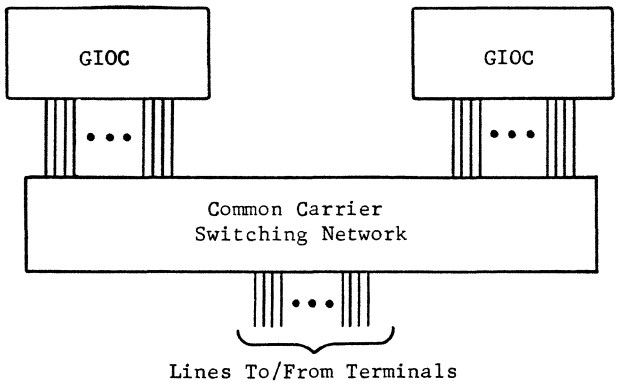


Figure 9. Normal Switching Network Connections

A distinction must be made between the number of terminals which have the capability of being connected to the system and the maximum number of terminals which may be connected at any given time. The former may be far larger than the latter. As an example, suppose that a system is capable of connecting and operating 200 terminals simultaneously. There could be 2,000 terminals which, at one time or another, are able to be connected to the system. However, as long as not more than 10 percent of these users wish to be connected to the system at any given moment, there is no problem or conflict.

Normally a terminal is connected to a GIOC through a common carrier switching network as has been discussed. This situation is indicated in Figure 9.

In certain special situations, the terminal may bypass a switching network. This might be done for special, high-speed communication lines, or for maintenance or supervisory entries into the system. These possibilities are shown in Figure 10.

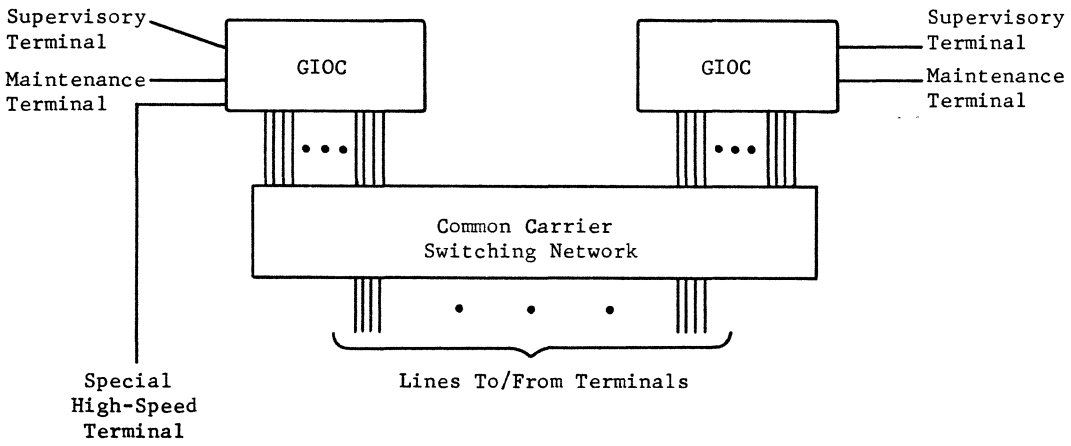


Figure 10. Bypassing the Switching Network

In some geographical areas or applications, it may be necessary for the system to communicate with terminals that are provided by more than one common carrier. If these two different common carrier companies are identified as A and B, Figure 11 shows a way to connect the terminals and switching networks to the GIOCs.

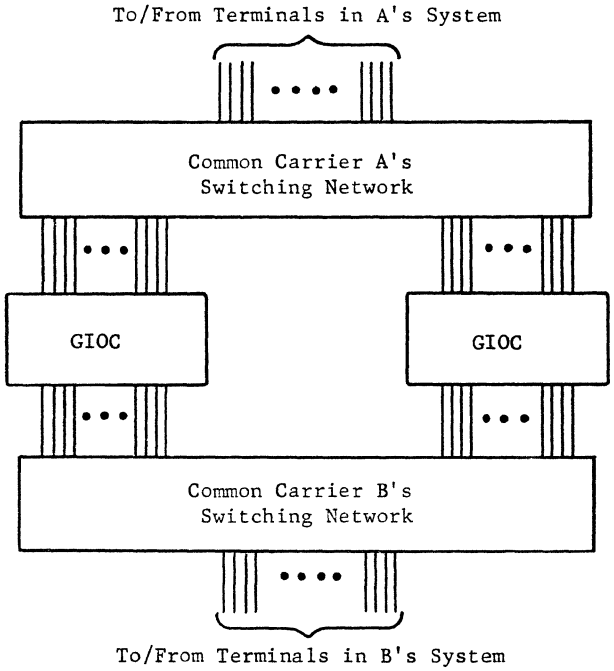


Figure 11. More Than One Switching Network

All of the preceding examples have assumed the existence of data sets between the terminal and the GIOC. However, this is not necessary. If the distance between the GIOC and the terminal is less than 50 feet, a teletypewriter may be connected to the GIOC without any intervening data sets. This is illustrated in Figure 12.

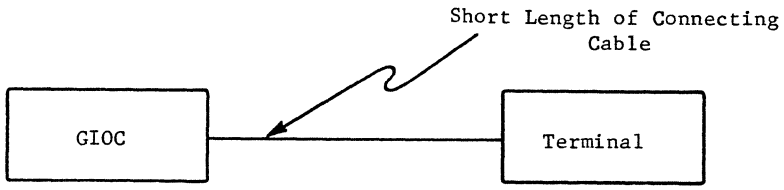


Figure 12. Connecting a Terminal Without Data Sets

6. PROCESSOR MODULE

Most of the advanced capabilities of the GE-645 which are seen by its users are provided by the processor. These include the extensive instruction repertoire, the comprehensive address modification capability, and unique paging and segmentation characteristics. The general properties of the processor are explained briefly in the first part of this chapter. This is followed by a more comprehensive discussion of those features which make the GE-645 especially well suited to operate at the heart of an information processing utility.

REGISTER DESCRIPTIONS

The numerous program-accessible processor registers are:

Quantity	Name	Length
1	Accumulator Register	36 bits
1	Quotient Register	36 bits
1	Exponent Register	8 bits
8	Index Registers	18 bits each
1	Memory Access Counter	24 bits
1	Instruction Counter	18 bits
1	Indicator Register	18 bits
1	Descriptor Base Register	29 bits
1	Procedure Base Register	18 bits
8	Address Base Registers	24 bits each

The Accumulator (A) Register is used:

- As an operand register for all classes of single-precision (36 bit) operations. For floating-point operations, the mantissa is in the A register and the exponent is in the Exponent Register.
- For address modification. Each half of the A register can hold an index value. The left half is called A Upper (AU) and the right half A Lower (AL).

The Quotient (Q) Register is used in the same ways as the A register, with the halves called QU and QL.

The AQ Register, which is the combined A and Q registers, is used as the operand register for all double-precision (72 bit) operations. For floating-point operations, the mantissa is in the A or AQ register and the exponent is in the Exponent Register.

The Exponent Register holds the exponent in floating-point operations with the mantissa in either the A or the AQ register.

The Memory Access Counter counts the number of memory cycles used. It is a method of measuring processor utilization. The counter may be set to an initial value which, when counted down to zero, causes the processor to be interrupted.

The Instruction Counter (IC) specifies the next instruction to be executed.

The Indicator Register contains a group of indicators that specify or hold certain processor states.

The various indicators are:

Zero
Negative
Carry
Overflow
Exponent Overflow
Exponent Underflow
Overflow Mask (can inhibit overflow recognition)
Tally Runout
Parity Error
Parity Mask (can inhibit parity error recognition)
Absolute Mode (use only IC for instruction fetching)
Not assigned.

The Descriptor Base Register (DBR), Procedure Base Register (PBR), and Address Base Registers (ABRn) are used in performing relative addressing. Their description is postponed until relative addressing is discussed later in this chapter.

CLASSES OF INSTRUCTIONS

For the GE-645, there are a possible 512 instructions of which 402 have been assigned. This leaves 110 operation codes for upward growth of the GE-645 line. A list of the assigned instructions is included at the end of this chapter.

The GE-645 instructions are listed and described according to operations for: data movement, arithmetic, logical, comparison, control, shifting, and special operations. Many of the GE-645 instructions are familiar to experienced programmers of large scale computers. In addition to the familiar ones, there are other instructions to facilitate: segmentation and paging, saving and restoring of registers, character handling, decision making, and list processing.

Data Movement Operations

These instructions move data between the system's core memory and the processor registers, indicators, and associative memory. They include half, single, and double-precision operations.

Arithmetic Operations

These instructions include fixed and floating point; half, single, and double-precision; and placement of results in registers or memory.

Logical Operations

These instructions include single and double-precision, and placement of results in registers or memory.

Comparison Operations

These instructions include logical, fixed, and floating-point comparisons on half, single, and double-precision operands. The results of a comparison are saved in certain indicators for later testing by control instructions.

Control Operations

These instructions include normal transfers, conditional transfers, and some special instructions which simultaneously transfer control and restore various registers.

Shifting Operations

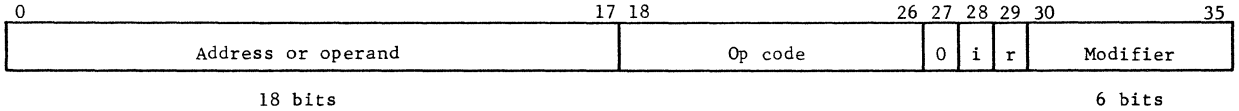
These are the normal instructions using the A, Q, and AQ registers.

Special Operations

An important instruction in this class in the Connect, which initiates all input/output operations and provides a method of communicating between processors. Another group of instructions enables one or two instructions to be repeated without having to continually fetch the instruction(s) from memory. Other instructions provide a way to execute one or two instructions which are not in the sequential string of instructions being executed, provide entries to the operating system, convert a binary number to its binary-coded-decimal equivalent, and convert a Gray Code word to a binary equivalent.

INSTRUCTION FORMAT

Most of the instructions have the following format. A few special instructions have a format adapted to their unique functions.



The address or operand field specifies information used in addressing an operand or contains the operand.

The operation code field specifies the operation to be performed and the registers to be used.

Bit 27 is not used and must be zero for upward compatibility with future systems.

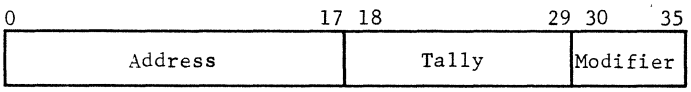
Bit 28 is used under certain conditions to inhibit a program interrupt. See Privilege and Modes of Operation, Page 45.

Bit 29 specifies how the address field is to be interpreted. See Base Registers, page 33.

The modifier field specifies the kind of address modification to be used. Possibilities include: (1) no modification, (2) which register to use, (3) indirect or not, and (4) several special-purpose modification methods that significantly increase the power of the instructions.

ADDRESS MODIFICATION

A comprehensive set of address modification features includes both modification of addresses by the contents of registers and multi-level indirect addressing. The general format of the indirect words is:



There are five classes of address modification: (1) register, (2) register then indirect, (3) indirect then register, (4) indirect then tally, and (5) indirect to pair. Four of these are described briefly below. The fifth, indirect to pair modification, is described under Indirect to Pair Modifiers, page 36.

Register Modification (R)

An effective address is produced by adding the contents of a modification register to the contents of the instruction's address field. Modification registers are the index registers, AU, AL, QU, QL, and IC. Direct operand modifiers 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) to obtain an indirect word, then conduct the modification specified in the indirect word. Address modification in an indirect word is specified in the same manner as for an instruction word.

Indirect Then Register (IR)

First obtain the indirect word using the original address, then conduct the modification specified in the indirect word. Upon completion of indirect addressing, perform the indicated register modification in the last IR word encountered.

Indirect Then Tally (IT)

Obtain the indirect word using the original address. Then perform one of the 13 possible sequences specified below. Many of these change the address and/or tally fields of the indirect word.

Mnemonic	Name
I	Indirect
AD	Add Delta (to address field)
SD	Subtract Delta (from address field)
DI	Decrement Address, Increment Tally
DIC	Decrement Address, Increment Tally, and Continue (using the modifier field of the indirect word)
ID	Increment Address, Decrement Tally
IDC	Increment Address, Decrement Tally, and Continue (using the modifier field of the indirect word)
CI	Character from Indirect
SC	Sequence Character
SCR	Sequence Character Reversed
F	Fault 1 (a processor fault)
F2	Fault 2 (a processor fault)
F3	Fault 3 (a processor fault)

FAULTS AND INTERRUPTS

The system must respond promptly when an event has occurred which needs servicing or when a hardware malfunction is detected. This fast response is obtained by having a unique pair of locations set aside for each fault and interrupt condition. The locations associated with faults are called the fault vector and the locations associated with interrupts are called the interrupt vector. Although a fault and an interrupt result in the same general system reaction, there is some difference between how faults and interrupts occur. Faults are conditions detected within a processor, while interrupts are usually conditions established by a GIOC or drum. However, a processor can also establish an interrupt condition.

The processor response is the same for a fault or an interrupt: the processor stops what it was doing and transfers to the location associated with that specific fault or interrupt. Multics-645/I has stored a pair of instructions here which save the processor's status and transfer to a routine that saves the processor's registers and services the fault or interrupt. At some point the interrupted routine will have the registers and processor status restored and then continue as though it had not been interrupted.

Many of the processor fault conditions are deliberately or inadvertently caused by the program and do not involve any hardware malfunction. Other faults are definitely caused by some hardware malfunction. In either case, a Multics-645/I routine is available to service the fault. All of the processor faults are listed as follows:

Fault Name

Startup
Execute
Trouble
Operation Not Completed
Lockup
Divide Check
Overflow
Parity
Illegal Memory Command
Illegal Descriptor
Illegal Procedure
635 Compatibility
635/645 Compatibility
Master Mode Entry 1
Master Mode Entry 2
Master Mode Entry 3
Master Mode Entry 4
Derail
Fault Tag 1
Fault Tag 2
Fault Tag 3
Directed Fault 0
Directed Fault 1
Directed Fault 2
Directed Fault 3
Directed Fault 4
Directed Fault 5
Directed Fault 6
Directed Fault 7

These are involved with the segmentation and paging processes.

Connect
Memory Access Counter Runout
Shutdown

RELATIVE ADDRESSING

The capability of the GE-645 processor to perform relative addressing is one of its most powerful features. It is this capability which allows it to perform the functions of paging and segmentation. The way in which these functions are accomplished is discussed on the following page.

In this discussion we will introduce the remaining registers of the processor. These are the descriptor base register, the eight address base registers, and the procedure base register.

Segmentation

Segmentation has been used in some form or other throughout the fields of computation and information processing for several years. In the GE-645, this familiar concept takes on added power and increased significance. In the past, program segments have generally consisted of subprograms and/or storage areas, and were combined into a single program by a loader prior to execution. This method of operation provided several important advantages over compiling or assembling entire programs prior to their execution. These advantages are preserved in the GE-645 segmentation, and other capabilities are added. Some of the added capabilities are:

- The ability for the same copy of a segment to be simultaneously shared by many users without making special prearrangements.
- The ability to allow individual segments to vary dynamically in size without influencing the addressing of other segments.
- The ability to address programs larger than available core memory.
- The ability for each segment to possess unique attributes to prevent misuse.

Significant among the advantages of segmentation is the facility it supplies for use of pure procedures. These are procedures which are not altered during their execution. As used in the GE-645, a segment can be identified as pure procedure and invoked by any user to perform its function on his own data. This provides two economies in the multi-user GE-645. First, only one copy of a pure procedure need exist to be shared by all its users; second, pure procedures constitute an important class of information that need never be rewritten to secondary storage.

It is often desirable to write a program that can operate on variable size data arrays. In previous computer systems it has been necessary to use complex overlaying techniques or to place an upper bound on the size of arrays that could be handled by such programs. In the GE-645, variable arrays can be assigned to separate segments so that each can vary in size from one up to 2^{18} words without any special provision being made by the computer users. (Segment size can be allowed to exceed 2^{18} words with the assistance of the Multics-645/I file system.)

A GE-645 program will consist of a variety of segments, each of which has a unique name. These names will be converted to a segment number by Multics-645/I at execution time. The user may view each of his segments as if it were stored in an independent core memory. Each segment has its own origin which can be addressed as location zero. The size of each segment may vary without affecting the addressing in other segments.

The words contained within a segment are addressed by an 18-bit segment address. If a program consists of several segments, the GE-645 refers to any word in any segment according to a segment number and a segment address. Figure 13 represents schematically the GE-645 memory topology. The individual segments of a program are shown as consisting of independent variable size storage arrays. In analogy to a rectangular coordinate system, it suggests that the location of any word may be specified by two coordinates: a segment number and a segment address.

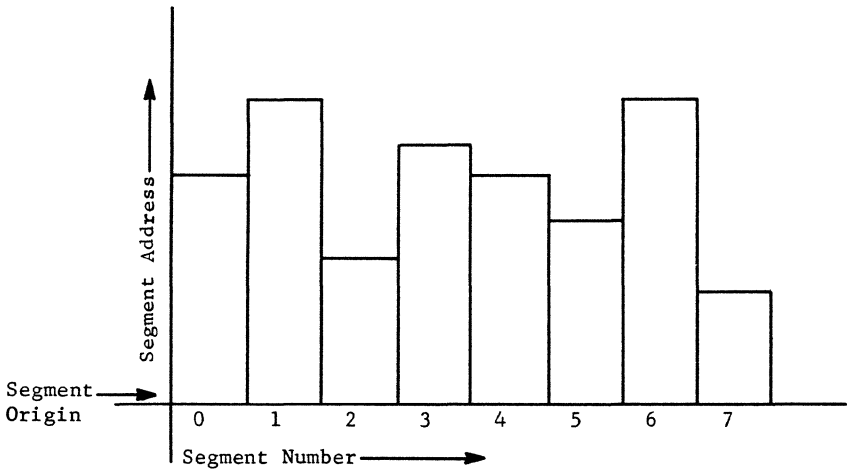


Figure 13. Two-Coordinate Addressing in the GE-645

DESCRIPTOR SEGMENT. A descriptor segment is the processor's means of relating program references to absolute memory locations. Multics-645/I provides a descriptor segment for each program. The descriptor segment contains one word called a segment descriptor word (SDW) for each segment being used by the program. Each SDW in the descriptor segment indicates the absolute location of the origin, and gives size and control information about the segment to which it corresponds (See Figure 14).

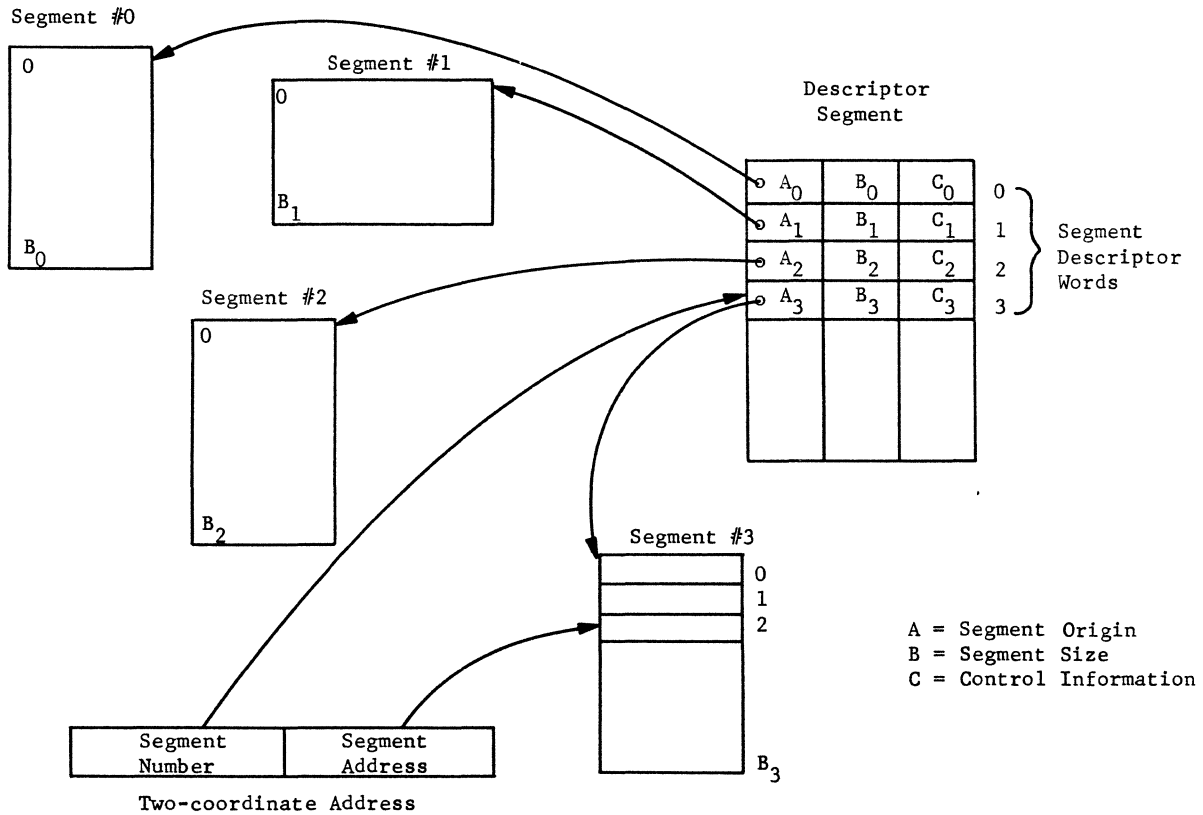


Figure 14. The Descriptor Segment Facilitates Two-Coordinate Addressing

The segment number used in locating a segment refers to the location of SDW in the descriptor segment. This SDW contains the segment origin. When a reference is made using a two-coordinate address, the processor retrieves the segment origin from the descriptor segment and adds the segment address to it. The result is the absolute address of the desired word. This is illustrated in Figure 14 and in numerous examples which follow. The segment address is compared to the segment size to verify that the reference is within the specified allowable address range of the segment.

Since the descriptor segment can be up to 2^{18} words long, the two-coordinate addressing of the GE-645 allows a program to reference up to 2^{18} segments.

BASE REGISTERS. The location of the descriptor segment for the process in execution is always known to the processor. Its origin and size are stored in a special high-speed register called the descriptor base register (DBR). (See Figure 15.) When control of the processor is switched from one user to another, the DBR is reloaded to point to the descriptor segment belonging to the new user. In this way all of the relative addressing information for all of the segments accessible to the program in execution are changed by simply changing the contents of one register.

The DBR, like the descriptor segment, is accessible only to Multics-645/I. This means it cannot be manipulated in order to gain unauthorized access to segments of another program. Also, storage belonging to another program cannot be accidentally damaged.

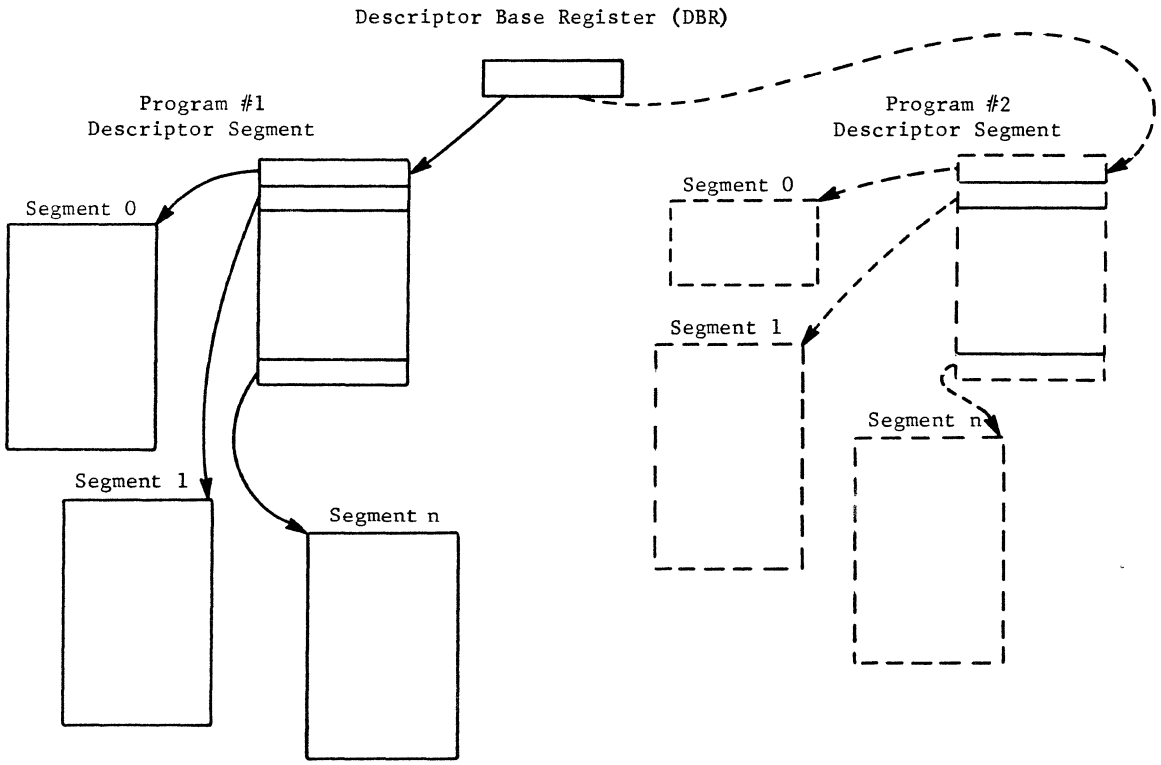


Figure 15. The DBR Determines which Segments the Processor May Access

A user may utilize segments in various ways depending upon his objectives. A simple way of using segments is to place instructions in one class of segments called procedure segments, and variable information in another class of segments called data segments. The GE-645 provides convenient methods for performing necessary references between such segments.

There are eight address base registers (ABR's) in each processor. They may be used to specify the segment number portion of a two-coordinate address as in Figure 16. If bit 29 of an instruction is set to 1, this signals the processor to interpret the three high-order bits of the address field as an address base register number. The ABR will contain the segment number to be addressed, and bits three through seventeen of the instruction (y in the figure) will contain the segment address.

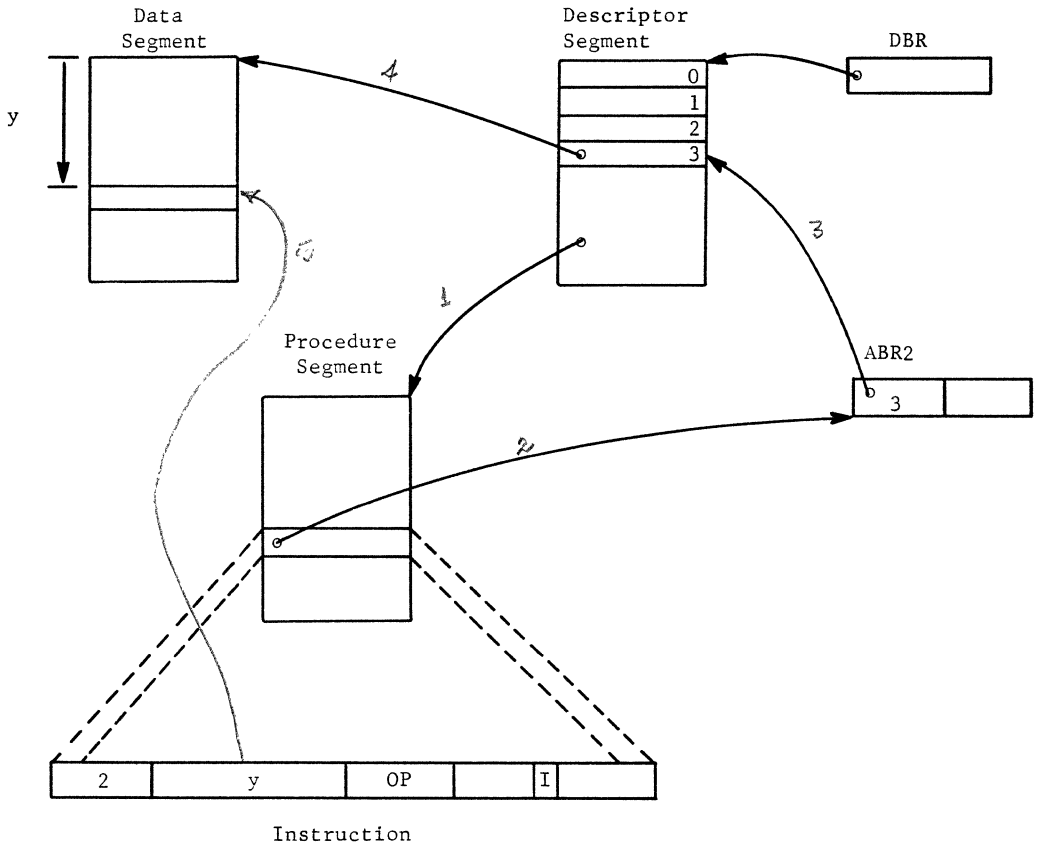


Figure 16. An ABR is Used in Specifying a Two-Coordinate Address to Refer to Another Segment

The processor is constantly made aware by the procedure base register (PBR) of the segment from which it is to retrieve instructions. The PBR functions exactly like an ABR except the PBR is never referenced explicitly in a user program. The processor automatically refers to the PBR in the execution of every instruction. The instruction counter (IC) contains the segment address and the PBR contains the segment number of the two-coordinate address of the next instruction. See Figure 17.

The PBR is automatically changed to contain the segment number of the target segment when a transfer between segments is performed. Figure 18 illustrates this process. In this case ABR2 contains the segment number which is placed in the PBR as part of the intersegment transfer operation.

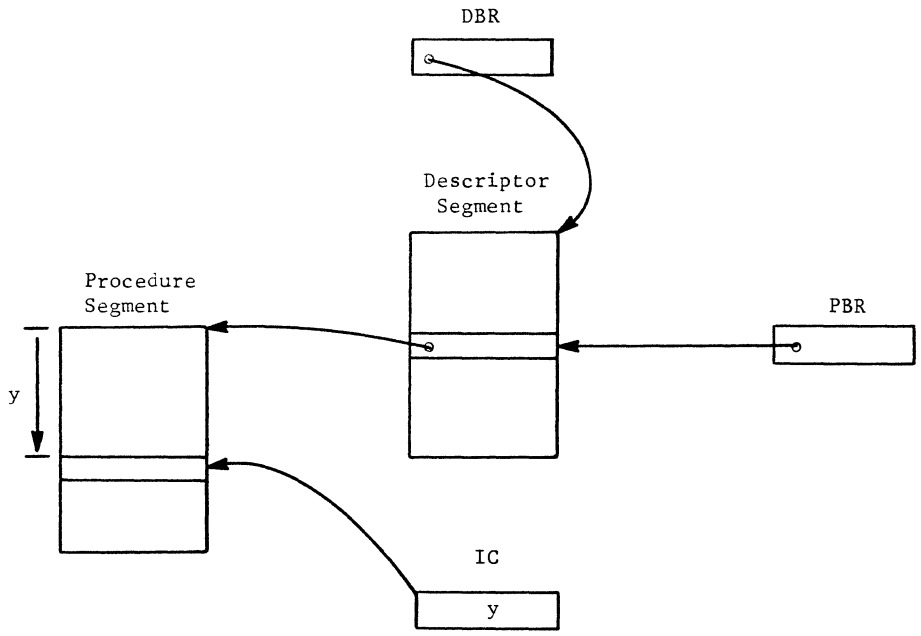


Figure 17. PBR Designates the Segment from which Instructions are Fetched

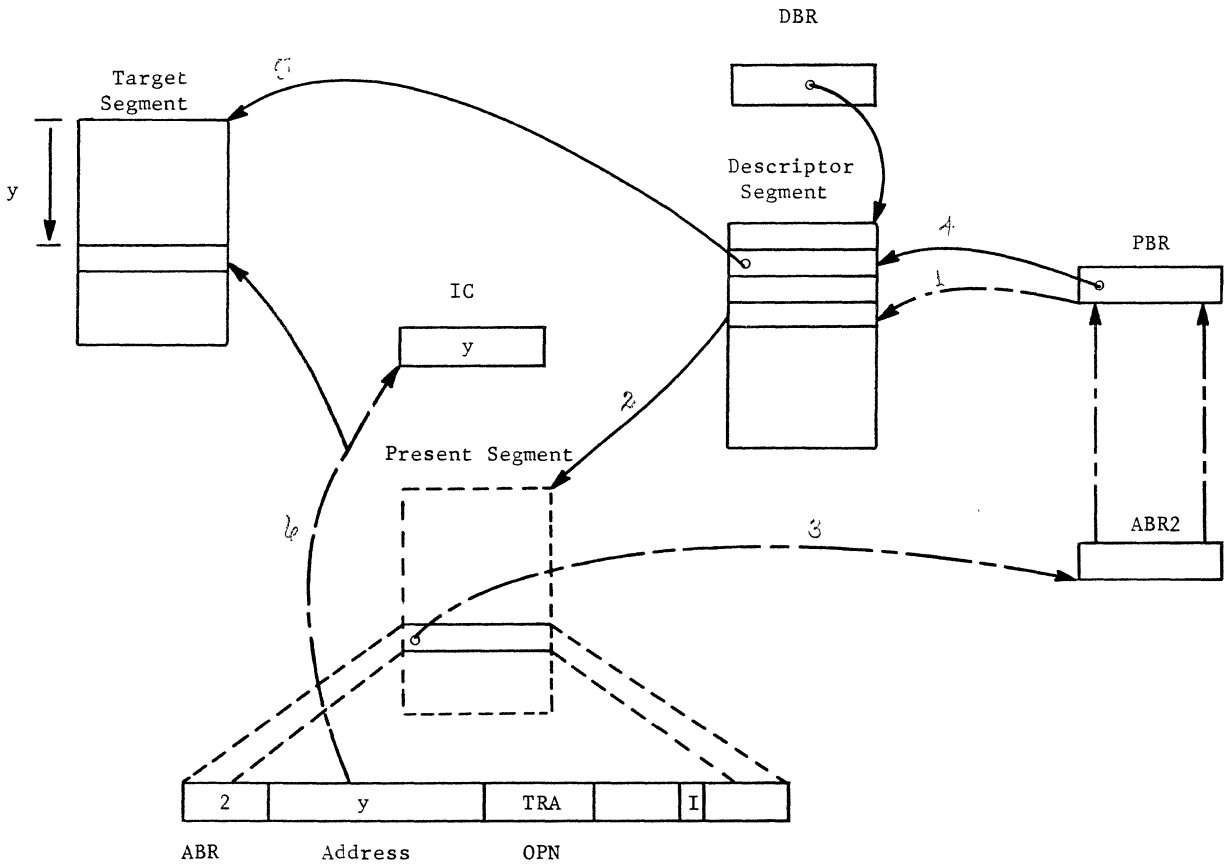


Figure 18. A Transfer Instruction Causes the Procedure Base Register to be Changed

INDIRECT TO PAIR MODIFIERS. An additional method of addressing between segments is provided in the GE-645. This is the indirect to pair (IP) address modifier. The IP modifier has two variations: indirect to segment modification (ITS), and indirect to base modification (ITB).

If an instruction refers indirectly to a word located in an even addressed location in core memory and if the indirect word contains the bit configuration corresponding to the ITS modifier in its modifier field, then the following occurs (See Figure 19).

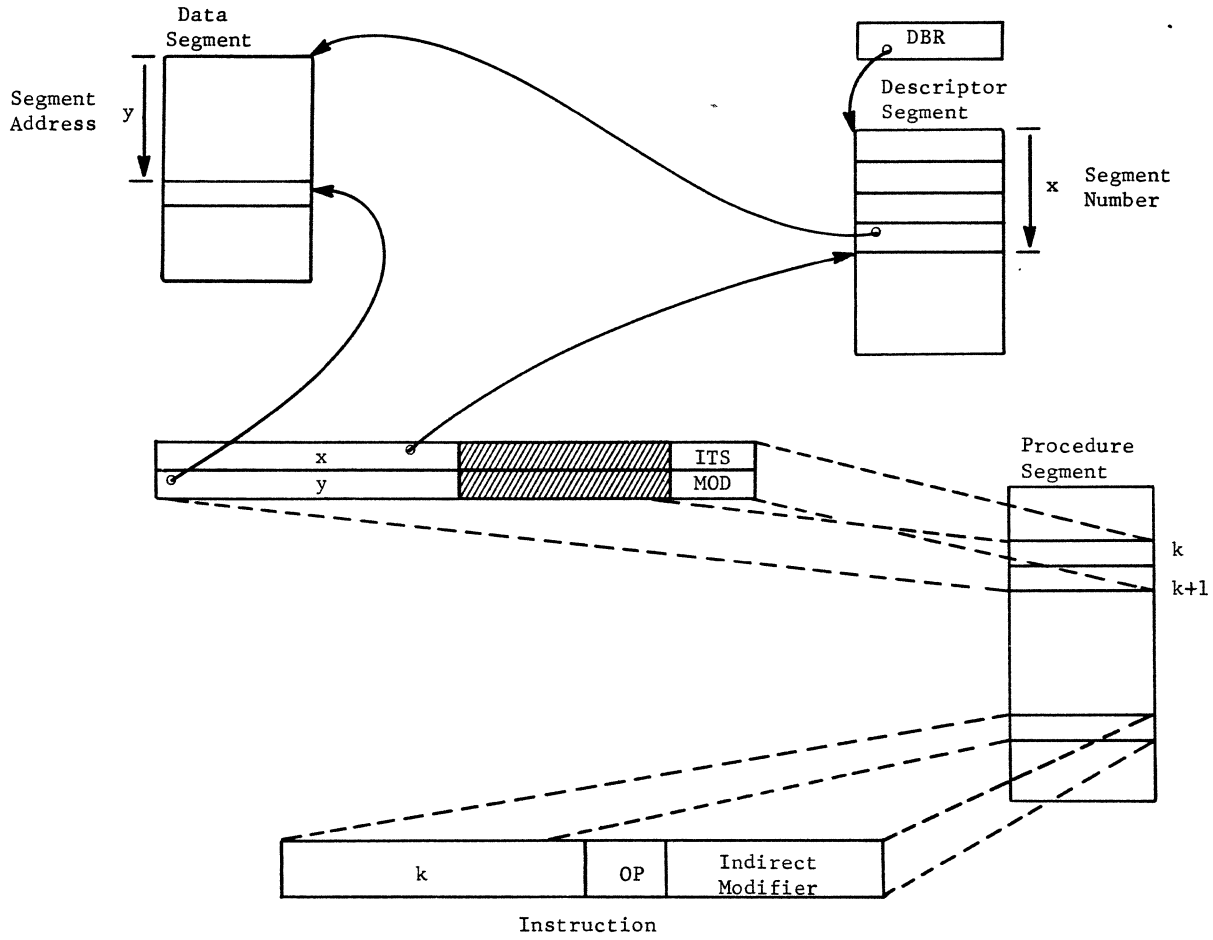


Figure 19. An ITS Pair is Used to Refer to Another Segment

1. Bits 0 through 17 of the indirect word are interpreted as a segment number.

2. Bits 0 through 17 of the word following the indirect word are interpreted as a segment address.

3. Bits 30 through 35 of the word following the indirect word are interpreted as a modifier to be applied to the segment address. Figure 19 illustrates the use of an ITS pair in performing a reference between segments.

ITB modification is similar to ITS modification in its use of indirect words. However, the interpretation of the even-odd pair is as follows (See Figure 20):

1. Bits 0 through 2 of the indirect word are interpreted as an ABR number.
2. The ABR contains a segment number to be used in relative addressing.
3. Bits 0 through 17 of the word following the indirect word are interpreted as a segment address.
4. Bits 30 through 35 of the odd word following the indirect word are interpreted as a modifier to be applied to the segment address.

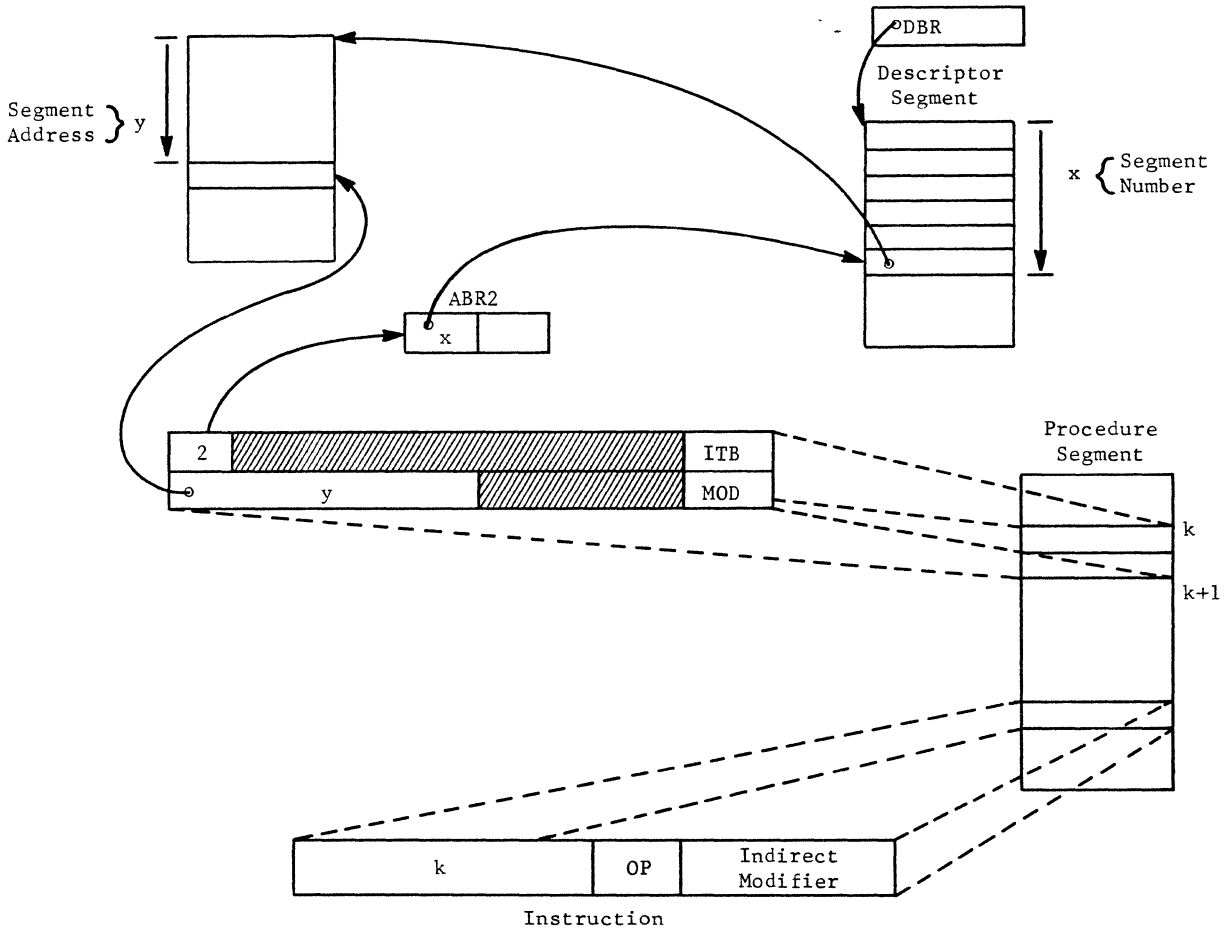


Figure 20. An ITB Pair is Used to Refer to Another Segment

BASE PAIRS. Address Base Registers (ABR's) may be used in pairs to provide a local origin within a segment. This is useful in various applications as an additional level of index-type modification. It is particularly useful in the maintenance of pushdown-popup stacks.

Figure 21 illustrates the use of a base pair. The base number specified in bits 0 through 2 of the instruction point to a base register which is designated as "internal" in its control field. The contents of this base register will be added to the address of the instruction providing a displacement from the segment origin. Three additional bits in the control field of the internal base register designate a second ABR containing the segment number of the target segment.

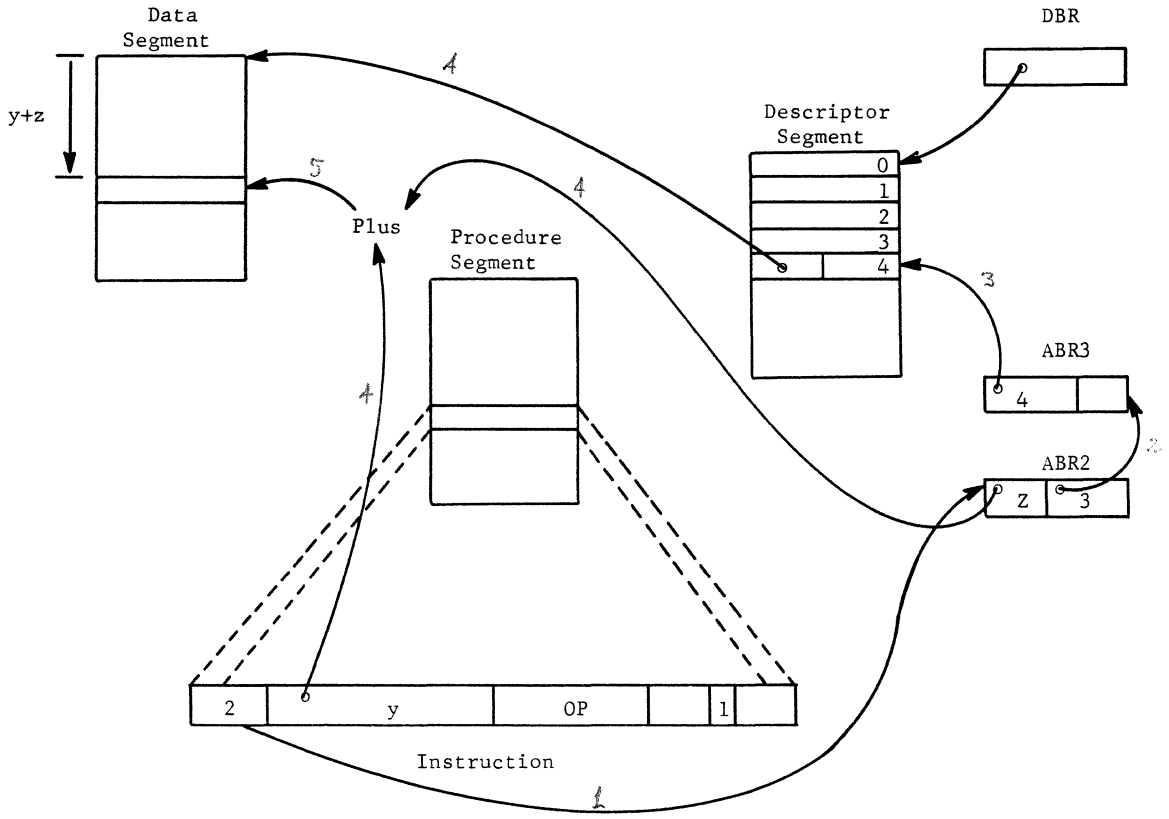


Figure 21. Operation of a Base Pair

SHARED SEGMENTS. Two or more programs may make common use of certain segments. For example, several users may share common data bases and pure procedures, such as compilers and utility routines. With the demonstration of appropriate authorization, Multics-645/I allows a user to access a common segment by placing a corresponding segment descriptor word (SDW) in his descriptor segment. Figure 22 shows the descriptor segments of three programs sharing a segment named XYZ. After Multics-645/I has entered the SDW's in the three descriptor segments, the three programs all refer to the common segment.

Figure 23 illustrates one use of shared segments. Here a single pure procedure operates on a separate data segment for every program in a uniprocessor system. Figure 24 illustrates the use of the same pure procedure segment by two processors in a dual processor configuration. In this case both processors could be simultaneously executing the same instructions on behalf of two different programs. And in general, any number of processors could be making simultaneous use of a pure procedure.

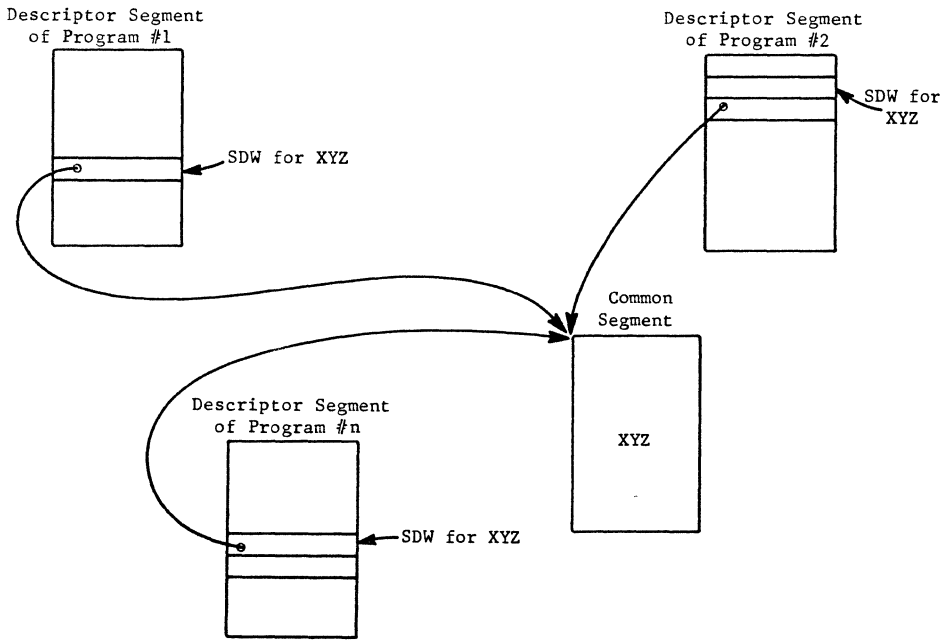


Figure 22. Several Descriptor Segments May Refer to a Common Segment

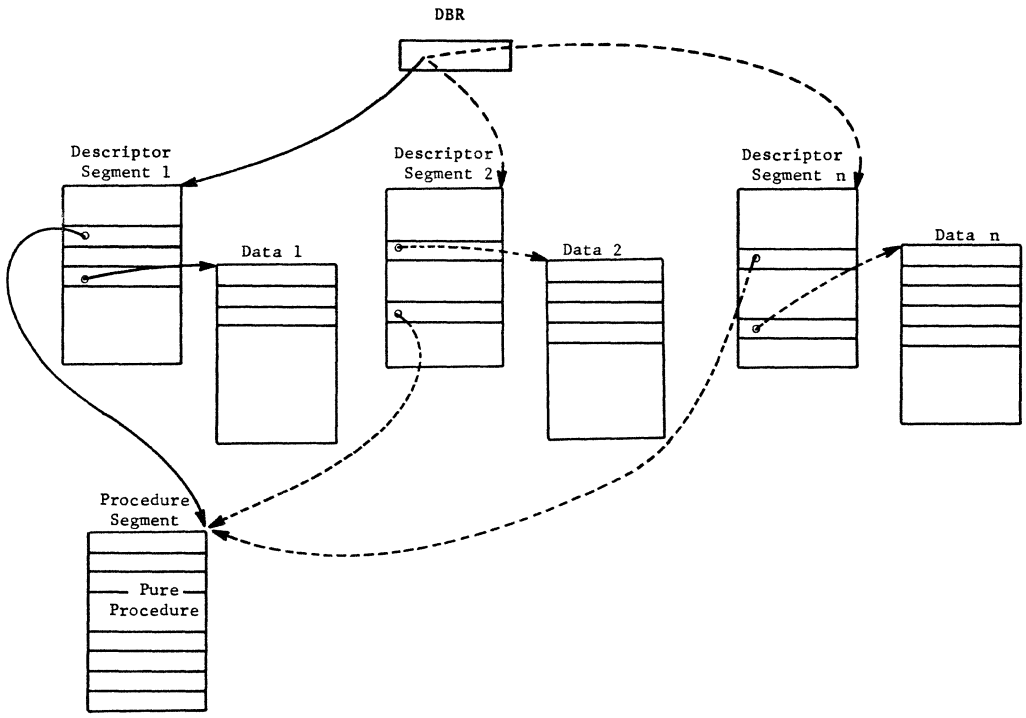


Figure 23. Common Pure Procedure Alternately Serves n Programs in a Uniprocessor Configuration

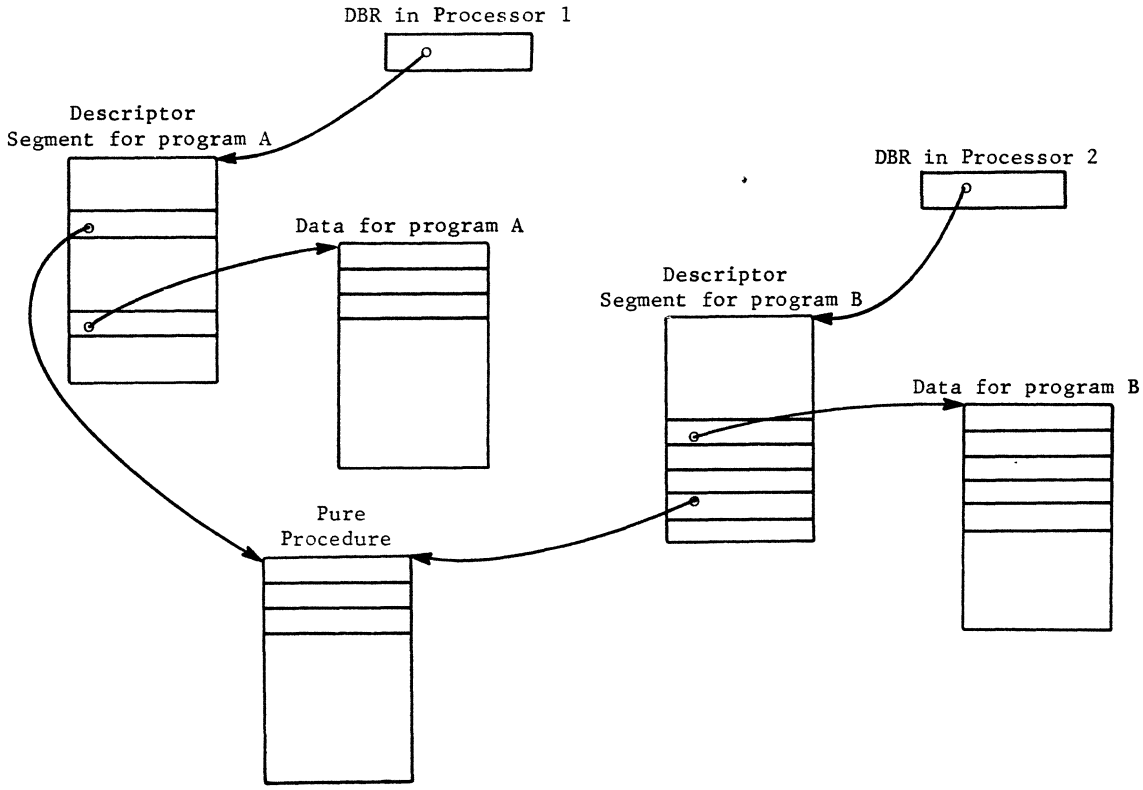


Figure 24. Common Pure Procedure Used Concurrently by Two Processors in Dual Configuration.

Paging

GE-645 users consider their programs to consist of named segments without regard for their absolute locations in core memory. In fact, only a small portion of most segments will be in core memory at one time. Instead, almost all of a segment will be stored in relatively inexpensive secondary storage devices in locations known to Multics-645/I. Only the relatively active portions of each segment will reside in core memory.

The GE-645 processor considers core memory to consist of blocks of 64 or 1024 words. Each block begins in an absolute address, which is 0 modulo 64 or 0 modulo 1024.

A segment may similarly consist of blocks of 64 or 1024 words called pages. Any pages of a segment may now be placed in any available core memory blocks of appropriate size. The GE-645 relative addressing capability will allow such pages to be addressed as if they are physically contiguous even though they are in widely scattered absolute locations. The above ideas are shown in Figure 25.

SEGMENT DESCRIPTOR WORD CONTROL FIELD. In describing segmentation, the origin and size fields of the SDW have been discussed. In describing paging, the control field of the SDW will be discussed. Two bits of the control field describe whether or not the corresponding segment is paged and if so whether the pages are 64 or 1024 words long.

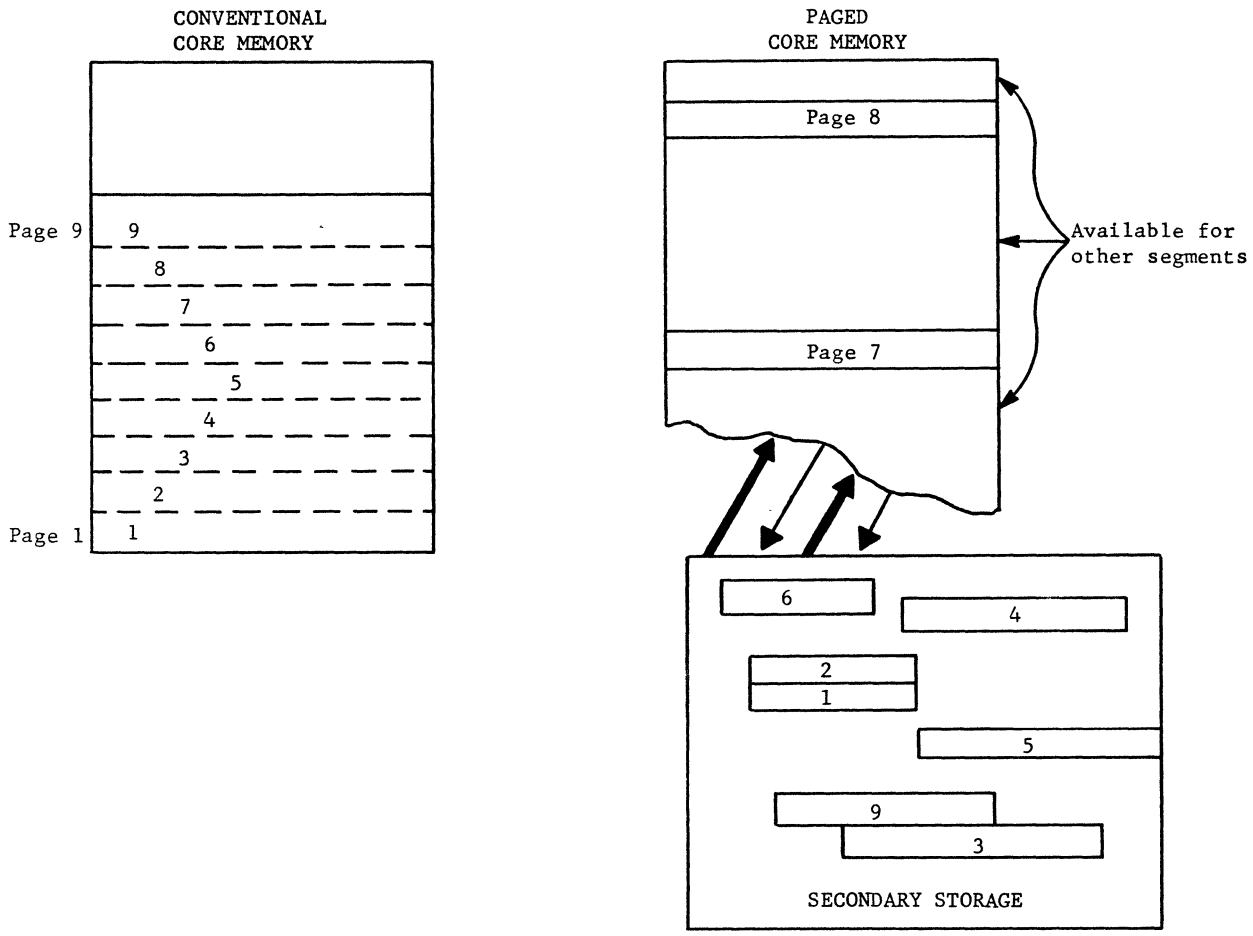


Figure 25. Frequently-Used Pages Occupy Any Available Blocks of Core Memory

If a segment is nonpaged, the complete segment is located in contiguous core memory addresses. Although segments are normally paged, nonpaged segments are allowed and can be accomodated by the GE-645 hardware and software.

In the usual case, a segment is paged. All of its pages are the same size, either 64 or 1024 words. The address field of its SDW specifies the origin of a page table, and not the segment origin as previously described.

Each word in a page table indicates the absolute location of the block in core memory to which the corresponding page is assigned; the first word locates the first page of the segment, the second word locates the second page, etc. The individual entries in the page table are called Page Descriptor Words (PDW's). The PDW's contain a control field as well as a page origin. (See Figure 26.)

SEGMENT ADDRESS PARTITIONING. When paging is used, each segment address is partitioned into two parts by the processor before it is used for address selection: a page number and a word number.

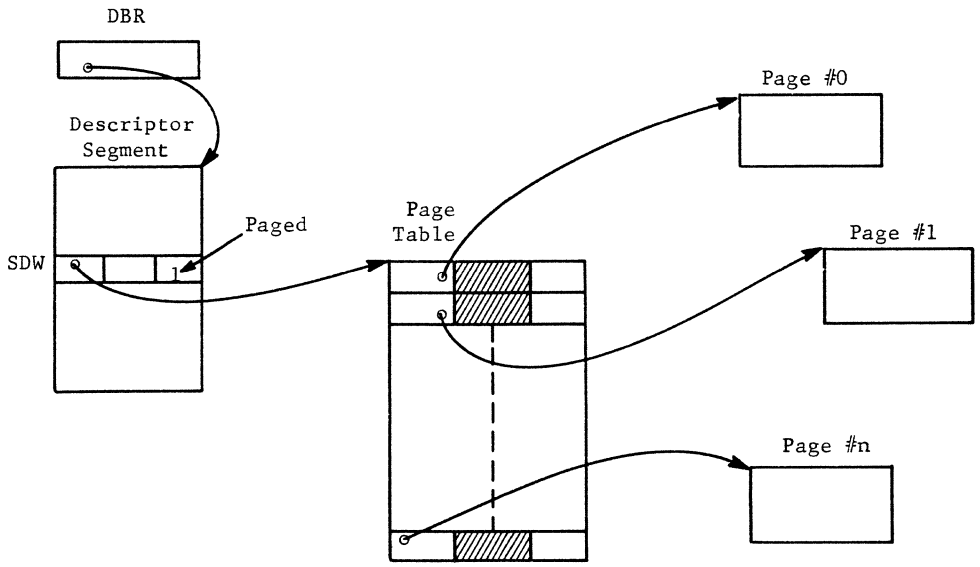


Figure 26. Page Table Words Specify Absolute Core Memory Origins

The processor partitions the segment address so that each entire page can be addressed by the word number field. This requires the word number to be 10 bits long for 1024-word pages and 6 bits long for 64-word pages. Figures 27 and 28 show segment address partitioning for two sizes of pages.

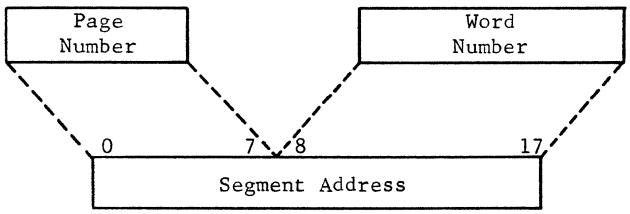


Figure 27. Partitioning of Segment Address for 1024-Word Pages

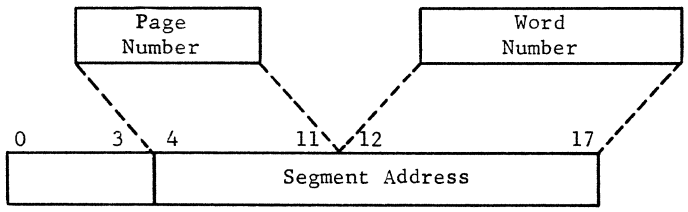


Figure 28. Partitioning of Segment Address for 64-Word Pages

PRODUCING THE ABSOLUTE ADDRESS. With this understanding of segmentation and paging, the actions of the processor in selecting an address from a page can be traced. See Figure 29. First, the segment number is determined and used to obtain a segment descriptor word (SDW) from the descriptor segment as described under segmentation. The SDW is now examined and found to refer to a paged segment. The segment address is partitioned into a page number and word number. The page number is added to the page table origin obtained from the SDW to locate the page descriptor word (PDW). Finally, the word number is added to the page origin obtained from the PDW to produce the absolute memory address of the desired word.

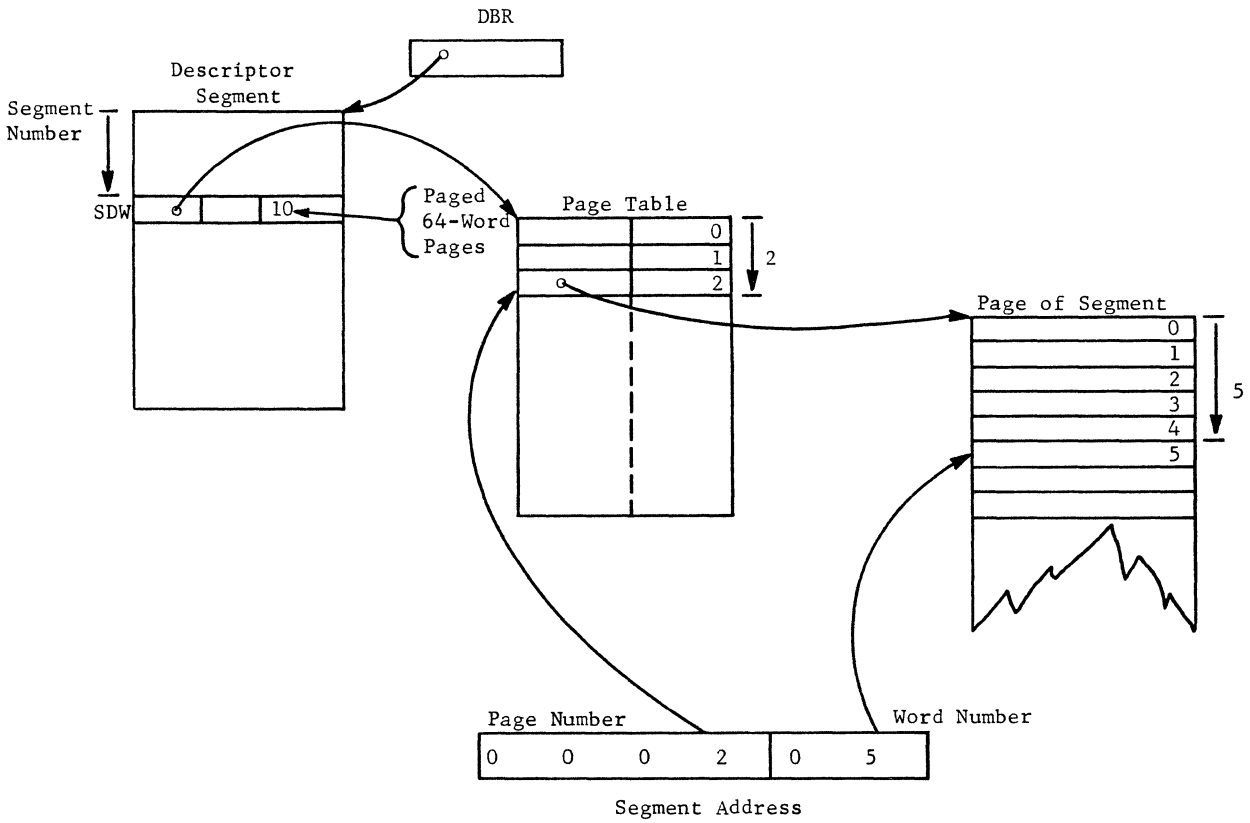


Figure 29. Selection of a Word From a Paged Segment

PAGED DESCRIPTOR SEGMENT. As previously mentioned most segments are paged. In fact, descriptor segments too can be paged. A two-bit control field in the descriptor base register specifies paging and page size in the same way as do the corresponding bits of the SDW's. If paging is specified, the descriptor segment has a page table which is used exactly as any other segment page table. See Figure 30.

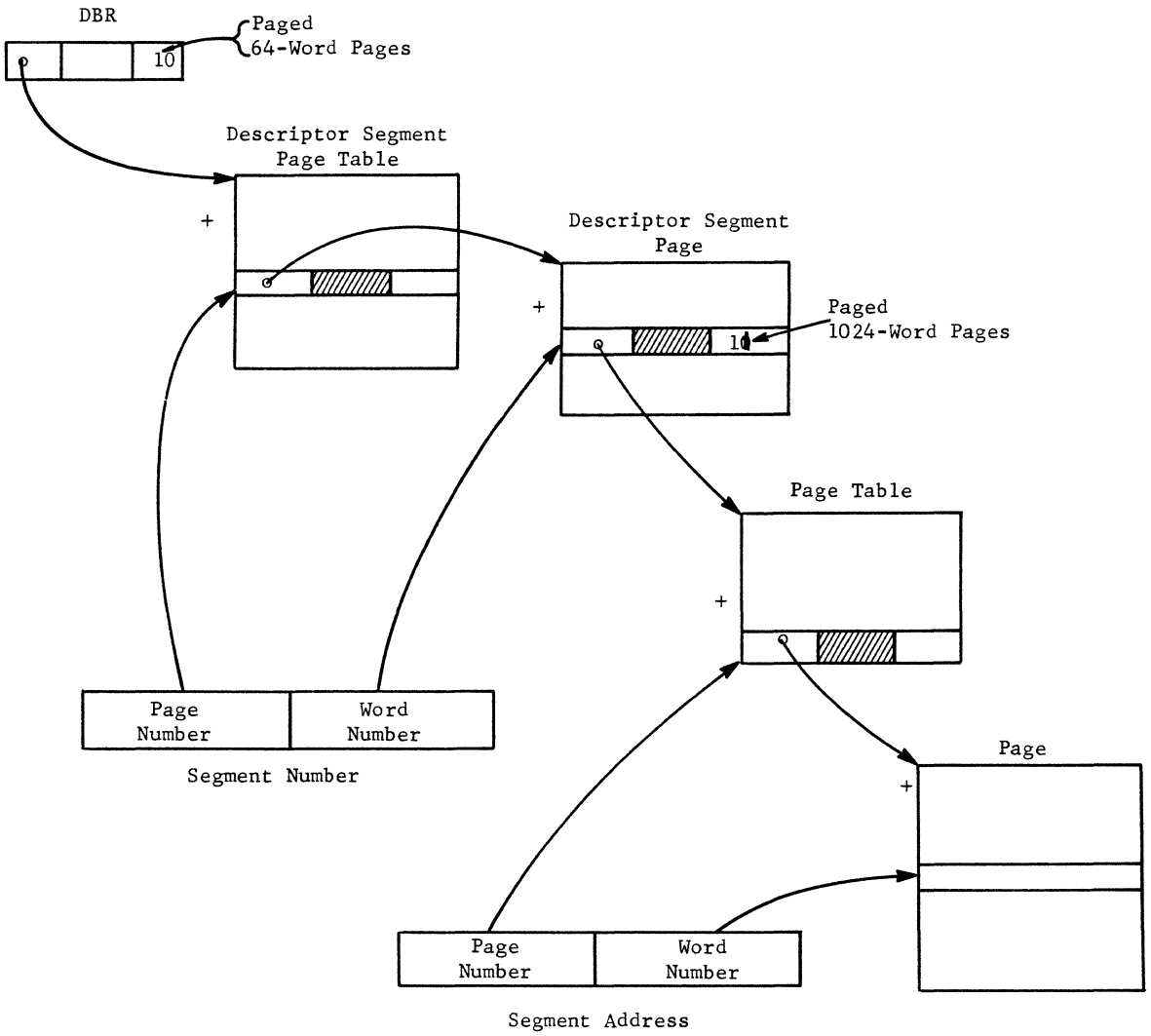


Figure 30. Selection of a Word From a Paged Segment
Using a Paged Descriptor Segment

ASSOCIATIVE MEMORY

The procedure just described involves several memory accesses to obtain the final operand. This sequence must be done once to locate the page, but only once. When the page's origin is determined, it is automatically saved in the processor's associative memory, along with the segment and page numbers. When a later access to the same segment number and page number is made, the absolute page origin may be provided immediately from the associative memory. Hence, repeated accesses to the descriptor segment page table, descriptor segment, and page table are unnecessary. Through incorporation of the associative memory, the powers of segmentation and paging are made available to GE-645 users while high processor performance is maintained.

PRIVILEGE AND MODES OF OPERATION

It has been said earlier in this chapter that the descriptor segment and descriptor base register are not directly accessible to the general user. Yet it is clear that there is a need for Multics-645/I to reference these. How then can general users be prohibited access to these while Multics-645/I is permitted access?

The answer is that the processor has three modes of operation: absolute, master, and slave. All instructions are available in absolute and master modes. Most, but not all, of the instructions are available in slave mode. General users are usually restricted to use of the slave mode and hence are prevented from executing any instructions which might allow them to do damage to programs of others or to Multics-645/I. Privileged instructions such as those which operate upon the descriptor base register, the system clocks, and input/output devices are available only in the absolute and master modes.

Whenever a fault occurs, the absolute mode is entered. This causes no trouble since faults place the processor in control of Multics-645/I which, like the hardware, can be assumed to be debugged. Absolute mode can also be entered from master mode, but the use of master mode is carefully guarded so that hazardous entries can be prevented. Instructions in the absolute mode can be inhibited from being interrupted. Only in the absolute mode may the absolute addresses of core memory be referenced by their true values. The relative addressing features associated with segmentation and paging can be activated optionally for referring to operands.

Whenever a processing unit is not in absolute mode, it will be in master or slave mode and use relative addressing in fetching instructions and operands. The mode is determined by a set of bits in the control field of the SDW of the segment from which the processor is fetching instructions. When the control field designates the segment to be a master procedure, the processor is in master mode. As such it is eligible to exercise all the absolute mode privileges except that of referring to absolute memory addresses. In addition to their other properties, master mode procedure segments, when entered from a slave mode instruction, may be entered only at location zero. This provides a method for Multics-645/I to verify the validity of such entries and prevent misuse of a master mode segment to the detriment of the system and its users.

If the segment is designated to be a slave procedure, interrupt inhibition indicated in individual instructions is ignored and execution of any privileged instructions results in a fault.

ACCESS CONTROL

In addition to the modes of the processor which determine certain privileges, access to segments can be further controlled. The foregoing has introduced the notion that segments can be declared to be master or slave procedures; they can also be declared to be data or "execute-only" procedures. These specifications are made as the third and fourth variations of the segment type portion of the SDW control field. If a segment is declared to be data, it cannot be accessed for instruction fetching. If a segment is declared to be an execute-only procedure, it can be accessed only for instruction fetching. A slave procedure can enter an execute-only procedure only at its relative location zero. This is the same restriction as for entering master mode segments. In all other ways, execute-only procedures are executed in the same way as slave procedures.

Access to all types of segments may be further qualified by the use of two additional and independent bits in the control field of a SDW. These are the write permit and master access bits. They may be used to prevent modification of a segment and to permit access to a segment only by master procedures. Either or both of these qualifications may be applied to any type of segment.

All pure procedures, regardless of their mode, should be consistent with their definition and use, and have write permission disallowed. In this way, attempted improper use by one user cannot diminish their usefulness to others.

Similarly, many users might find the general availability of certain data, e.g. tables of physical constants, to be valuable for their common use. However, if a single user stored an incorrect value in a table, its potential usefulness to all would be destroyed. Such eventualities can be eliminated by proper use of the write-permit bit and the declarations available through Multics-645/I which control setting of the write-permit bit.

The features of the three processor modes of operation are summarized in the following chart.

	Modes		
	Absolute	Master	Slave
Can privileged instructions be executed?	Yes	Yes	No
Can bit 28 be set to inhibit an interrupt?	Yes	Yes	No
Type of Address for instruction fetch	Absolute	Relative	Relative
Type of Address for operand fetch	Abosulte or Relative	Relative	Relative
Restrictions on access to other segments and pages?	None	None	Some

List of GE-645 Instructions

The following are the GE-645 instructions:

DATA MOVEMENT INSTRUCTIONS

Load

Load Accumulator
Load Quotient Register
Load A-Q Register
Load Index Register n
Load Index Register n from Lower
Load Complement into Accumulator

Load (continued)

Load Complement into Quotient Register
Load Complement into A-Q Register
Load Complement into Index Register n
Floating Load
Double Precision Floating Load
Load Exponent Register
Load Indicator Register
Load Memory Access Counter
Load Registers
Load Address Base Registers
Load Address Base Register n
Load Descriptor Base Register
Load Control Field
Effective Address to Accumulator
Effective Address to Quotient Register
Effective Address to Index Register n
Effective Address to Base Register n
Effective Address to Base Pair n

Store

Store Accumulator
Store Quotient Register
Store A-Q Register
Store Index Register n
Store Index Register n in Lower
Store Zero
Store 6-bit Characters of Accumulator
Store 6-bit Characters of Quotient Register
Store 9-bit Characters of Accumulator
Store 9-bit Characters of Quotient Register
Floating Store
Double-Precision Floating Store
Store Exponent Register
Store Indicator Register
Store Memory Access Counter
Store Instruction Counter +1 and Indicators
Store Instruction Counter +2
Store Registers
Store Address Base Registers
Store Address Base Register n
Store Descriptor Base Register
Store Pair in Address Base Register n
Store Associative Memory
Store Associative Memory Zero

ARITHMETIC INSTRUCTIONS

Fixed-Point Addition

Add to Accumulator
Add to Quotient Register
Add to A-Q Register
Add to Index Register n

Fixed-Point Addition (continued)

Add Logical to Accumulator
Add Logical to Quotient Register
Add Logical to A-Q Register
Add Logical to Index Register n
Add Accumulator to Storage
Add Quotient Register to Storage
Add Index Register n to Storage
Add with Carry to Accumulator
Add with Carry to Quotient Register
Add Low to A-Q Register
Add One to Storage
Add to Address Base Register n

Fixed-Point Subtraction

Subtract from Accumulator
Subtract from Quotient Register
Subtract from A-Q Register
Subtract from Index Register n
Subtract Logical from Accumulator
Subtract Logical from Quotient Register
Subtract Logical from A-Q Register
Subtract Logical from Index Register n
Subtract Accumulator from Storage
Subtract Quotient Register from Storage
Subtract Index Register n from Storage
Subtract with Carry from Accumulator
Subtract with Carry from Quotient Register

Fixed-Point Multiplication and Division

Multiply Integer
Multiply Fraction
Divide Integer
Divide Fraction

Floating-Point Arithmetic Instructions

Floating Add
Double-Precision Floating Add
Unnormalized Floating Add
Double-Precision Unnormalized Floating Add
Add to Exponent Register
Floating Subtract
Double-Precision Floating Subtract
Unnormalized Floating Subtract
Double-Precision Unnormalized Floating Subtract
Floating Multiply
Double-Precision Floating Multiply
Unnormalized Floating Multiply
Double-Precision Unnormalized Floating Multiply
Floating Divide
Double-Precision Floating Divide
Floating Divide Inverted
Double-Precision Floating Divide Inverted

Miscellaneous Arithmetic Instructions

Negate Accumulator
Negate Long
Floating Negate
Floating Normalize

LOGICAL INSTRUCTIONS

AND

AND to Accumulator
AND to Quotient Register
AND to A-Q Register
AND to Index Register n
AND to Storage from Accumulator
AND to Storage from Quotient Register
AND to Storage from Index Register n

OR

OR to Accumulator
OR to Quotient Register
OR to A-Q Register
OR to Index Register n
OR to Storage from Accumulator
OR to Storage from Quotient Register
OR to Storage from Index Register n

EXCLUSIVE OR

EXCLUSIVE OR to Accumulator
EXCLUSIVE OR to Quotient Register
EXCLUSIVE OR to A-Q Register
EXCLUSIVE OR to Index Register
EXCLUSIVE OR to Storage from Accumulator
EXCLUSIVE OR to Storage from Quotient Register
EXCLUSIVE OR to Storage from Index Register n

COMPARISON INSTRUCTIONS

Compare With Accumulator
Compare With Quotient Register
Compare With A-Q Register
Compare With Index Register n
Compare AND with Accumulator
Comparative AND with Quotient Register
Comparative AND A-Q Register
Comparative AND with Index Register n
Comparative NOT with Accumulator
Comparative NOT with Quotient Register
Comparative NOT with A-Q Register
Comparative NOT with Index Register n
Compare Masked
Compare Magnitude
Compare With Limits
Set Zero and Negative Indicators from Storage

COMPARISON INSTRUCTIONS (continued)

Floating Compare
Floating Compare Magnitude
Double-Precision Floating Compare
Double-Precision Floating Compare Magnitude
Floating Set Zero and Negative Indicators

CONTROL INSTRUCTIONS

Transfer Unconditionally
Transfer on Zero
Transfer on Non-Zero
Transfer on Plus
Transfer on Minus
Transfer on Carry
Transfer on No Carry
Transfer on Overflow
Transfer on Tally Runout Indicator Off
Transfer on Exponent Overflow
Transfer on Exponent Underflow
Transfer and Set Slave
Transfer and Set Index Register n
Transfer and Set Base Register n

SHIFTING INSTRUCTIONS

Accumulator Left Shift
Accumulator Right Shift
Accumulator Left Rotate
Accumulator Right Logical
Quotient Register Left Shift
Quotient Register Right Shift
Quotient Register Left Rotate
Quotient Register Right Logical
Long Left Shift
Long Right Shift
Long Left Rotate
Long Right Logical

SPECIAL INSTRUCTIONS

Connect I/O Channel
Load Alarm Clock
Read Calendar Clock
Set Memory Module Interrupt Cells
Read Memory Module Mask Register
Set Memory Module Mask Register
Clear Associative Memory
Store Control Unit
Restore Control Unit
Store Control Double
Return Control Double
Return
No Operation
Master Mode Entry 1
Master Mode Entry 2

SPECIAL INSTRUCTIONS (Continued)

Master Mode Entry 3
Master Mode Entry 4
Repeat
Repeat Double
Repeat Link
Execute
Execute Double
Binary to Binary Coded Decimal
Grey to Binary
Derail
Delay Until Interrupt Signal
Read Switches

GE-625/635 INSTRUCTIONS NOT IN GE-645

Load Base Address Register
Store Base Address Register

The base address register of the GE-625/635 has been superseded by the relative addressing hardware of the GE-645.

7. GENERALIZED INPUT/OUTPUT CONTROLLER

The generalized input/output controller (GIOC) is a free standing unit which consists of a controller and several types of modular operational units called adapters. The adapters contain data and list channels which are addressed by the program. The GIOC controls data transfers between input/output devices and memory modules of the GE-645. In controlling data transfers the GIOC receives commands from the software to transfer data to or from memory and provides status information to memory. Figure 31 illustrates the configuration of a GIOC with two adapters.

In transmission from the GIOC, a data set, as shown in the figure, converts a bit pattern from the adapter into signals acceptable to the common carrier communication lines. Another data set then converts the signal back to a form acceptable to the remote terminal.

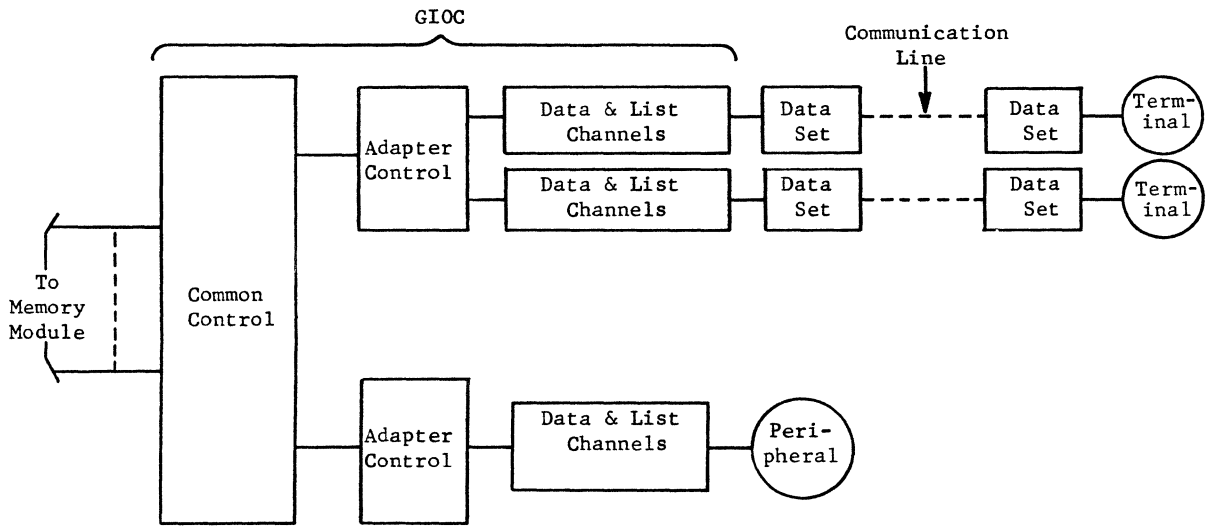


Figure 31. Functional Organization of the GIOC

Data can be transferred to and from memory in a number of different formats. Transfer is under control of the software and depends upon the type of adapter. A 36-bit word can contain one or six characters of up to 6 bits in length; or it can contain one or four characters of up to 9 bits in length.

CONTROLLER

The controller permits the channels to have time-shared access to the memory modules. In performing this function, the controller:

Performs service in response to requests from the adapters.
Controls the information transfer between the adapters and memory.
Controls the transfer of status from the adapters to the controller status channels.
Selects the proper memory port according to the memory address.
Senses Connect commands.
Issues program interrupts.
Contains the hardware forming the status and connect channels.
Contains a diagnostic command register.

The controller has a minimum of one and a maximum of eight identical ports to access memory. One memory module can be attached to each GIOC controller port. Cables connecting the memory and GIOC modules contain 36 input data lines, 36 output data lines, and a number of special function and control lines.

The controller has three connect channels to control the transfer of commands from memory to the adapter channels. The connect channels are permanently assigned channel numbers 8, 9, and 10.

The controller has at least four and may have as many as eight status channels to serve as temporary storage for status information to be stored in memory from the adapter channels. Status channels are permanently assigned channel numbers 0-7. Channel 0 has the highest priority and is reserved for use in indicating emergencies. When a condition requiring status storage is detected, the control word currently in use determines the status channel to be activated. The activated status channel then requests service to store the status in memory.

CHANNELS

The GIOC has four types of channels. Some are direct and some are indirect.

Data channels
List channels
Connect channels
Status channels

Data Channels

Data channels control data transfers between the memory modules and the peripheral subsystems or communication lines. There is one data channel for each peripheral subsystem or communication line connected to the GIOC. Each data channel provides the necessary data buffering and hardware to control the attached peripheral subsystem or communication line. Because of the diverse requirements for interfaces and buffering, there are a number of different types of data channels. All data channels are half-duplex; two data channels are required for full-duplex operation.

List Channels

List channels obtain data control words for subsequent control of the operation of data channels. The software arranges the data control words in sequential list form in memory. A separate list channel is associated with each data channel.

Connect Channels

Connect channels control the transfer of channel command words to data channels. The software arranges the channel command words in sequential list form in memory. A Connect command issued by the software initiates the operation of a connect channel. A connect channel can transfer channel command words to any data channel.

Status Channels

Status channels control the transfer of status information from data channels to memory. The status channels arrange the status words in sequential list form in memory. A status channel can transfer status information to memory from any data channel, list channel, or connect channel.

Direct Channels

The operation of each channel is controlled by a control word which provides a memory address to be used in communication between the channel and memory. A channel is called a direct channel when its control word can reside in the channel. Only data channels can be direct channels. Adapters with direct channels require only one memory access for each 12 characters of data transferred. Character transfer rates of at least 400,000 characters per second can be achieved by each direct channel. Because of this, direct channels are used for high-speed peripheral equipment such as discs and magnetic tapes.

Indirect Channels

Control words also control the operation of indirect channels. However, the control word does not reside in the channel, but resides in a mailbox in core memory. The channels are called indirect because they supply a channel number instead of an address. The controller converts the channel number into the address of the mailbox and then obtains the control word from the mailbox. Adapters with indirect channels require three memory accesses for each character or word of data transferred. The transfer rate is sufficient to keep communication lines and common peripheral equipment such as card readers, card punches, and printers operating at full speed.

ADAPTERS

An adapter is a bridge between an input/output device and the electronics of the GIOC controller. The adapters are independent units which can be installed to fit the exact requirements of the peripheral and remote terminal equipment. Each adapter contains one or more identical data channels to provide the necessary interface for attaching a particular type of peripheral or communication line. Each data channel has a list channel associated with it to assist in controlling the operation. Each data channel is odd-numbered, and the associated list channel has the next lower even number. In most cases, channels in an adapter are card-pull options.

All of the input/output devices listed in Chapter 10 can be connected to the GIOC. Depending upon the speed of data transfer, a data channel can transfer data by character, word, or double word. Transfer rates from communication lines may vary from 45 bits per second to 40,800 bits per second. Speeds higher than 40,800 bits per second require special hardware not in the basic GIOC.

The peripheral adapters operate with the standard Computer Equipment Department common peripheral interface. The communication adapters operate with an interface designed particularly for the type of remote terminal equipment and method of communication. For example, communication adapters with speeds less than 3000 bits per second operate with the EIA RS-232A interface. All adapter channels are half duplex. The modular construction of the adapters allows them to be easily added or replaced.

Peripheral Adapters

HIGH DATA RATE PERIPHERAL ADAPTER. This adapter controls one high-speed peripheral subsystem using a direct channel operating via the common peripheral interface. The maximum throughput for this channel is 400,000 characters per second.

STANDARD DATA RATE PERIPHERAL ADAPTER. This adapter controls up to six standard speed peripheral subsystems using six indirect channels. The maximum throughput for each channel is 10,000 characters per second.

Communication Adapters

TELETYPEWRITER ADAPTER. This adapter terminates up to 32 half-duplex teletypewriter lines. Plug selection provides for any of the standard bit rates between 45 and 200 bits per second and for 5-, 6-, 7-, or 8-level codes. All channels within the adapter operate at the same speed and code level. The channels can interface with the Bell Systems 103 Series data sets, and other similar modems. This adapter adds start/stop bits on outgoing characters and strips them from incoming characters.

An increased level of message assurance is available through the echoplex feature. This feature assures the terminal operator that the character on his visual output is what the system has received. It is accomplished by having the data channel receive a character and then retransmit it immediately to the sending station. It is this retransmitted character that appears as the visual output of the teletypewriter or other terminal.

SYNCHRONOUS SERIAL CHARACTER ADAPTER. This adapter terminates up to three half-duplex, voice-grade lines. The bit rate of each channel within the adapter is dependent on timing signals from the attached data set and is, therefore, independent on each channel. These adapters can interface with the Bell System 201 Series data sets or similar modems. Plug selection can choose 5-, 6-, 7-, or 8-level codes. A synchronization character establishes synchronization between the sending and receiving points before the actual data transfer. The character is plug selectable by channel.

ASYNCHRONOUS SERIAL CHARACTER ADAPTER. This adapter terminates from one to three half-duplex, voice-grade lines. Plug selection provides for any of the standard bit rates between 150 and 2400 bits per second and for 5-, 6-, 7-, or 8-level codes. The channels can interface with the Bell 103 and 202 Series data sets or similar modems. All three channels within this adapter must operate at the same speed and code level. Stop/start bits are added to outgoing characters and stripped from the incoming characters by the GIOC.

DIALING ADAPTER. This adapter provides up to eight dialing channels. The dialing rate depends on timing from the attached automatic-call unit and is independent in each channel. The channels can interface with the Bell System Series 801 automatic call unit or similar units. One dialing channel is required for each data channel that terminates a dial-out line.

INPUT/OUTPUT OPERATION

The GIOC and the software communicate by a combination of the Connect command, control words, program interrupts, and status-word queues. In general terms, the series of events for an input or output operation is as follows:

1. Various control words are set up in memory by the operating system.
2. The operating system issues a Connect command.
3. The GIOC fetches control information from memory to find out what to do.
4. The data transfer takes place. Individual control words are used to control each string of data flow.
5. When the data transfer is done, the GIOC obtains or builds status information and stores it in a status-word queue in memory.
6. The GIOC sets an interrupt cell in a memory module.
7. The memory module sends a signal to a processor.
8. The processor interrupts what it was doing and transfers control to a routine that services an interrupt.

CONTROL WORDS AND MAILBOXES

The GIOC obtains control information from memory modules by using mailboxes. Mailboxes are contiguous locations in memory. They contain control information generated by Multics-645/I for use by the GIOC in performing its functions. Each mailbox consists of a pair of 36-bit words.

There is a mailbox for each status and connect channel of the GIOC controller and a mailbox for each list and data channel of an adapter. Two mailboxes are reserved for diagnostic purposes. The GIOC moves control words, as required, from lists in memory into specific mailbox locations to control the activity of the channels. These control words direct all activity relating to the channels from the time the I/O activity is initiated until a terminate condition occurs and the channel returns to an idle status awaiting a new activity. The location of the first mailbox is

specified by a base address. The starting location must be modulo a power of two which is greater than twice the highest used channel number. Four types of control words are stored in mailboxes. These are:

Command Pointer Word (CPW)
List Pointer Word (LPW)
Data Control Word (DCW)
Status Control Word (SCW)

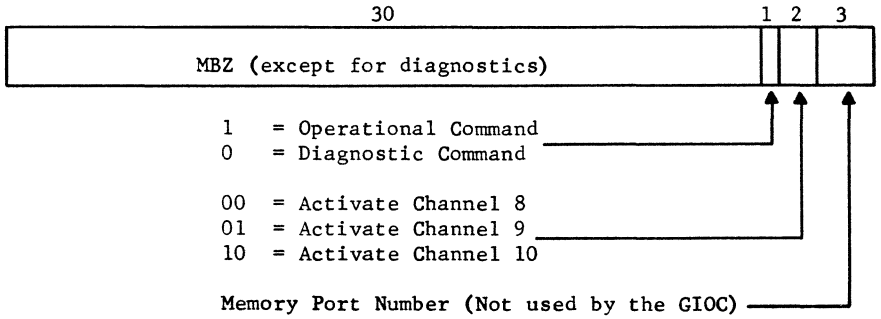
Additional control words are used by the GIOC, but are not stored in the mailboxes. These are:

Connect Operand Word
Channel Command Word (CCW)

The definition and format of each of these control words follow. The words are presented in the same order that they are used by the GIOC during an input/output operation. In the diagrams, word fields which contain all zeros are labeled MBZ (must be zero). The number above each field indicates that number of bit positions in the field.

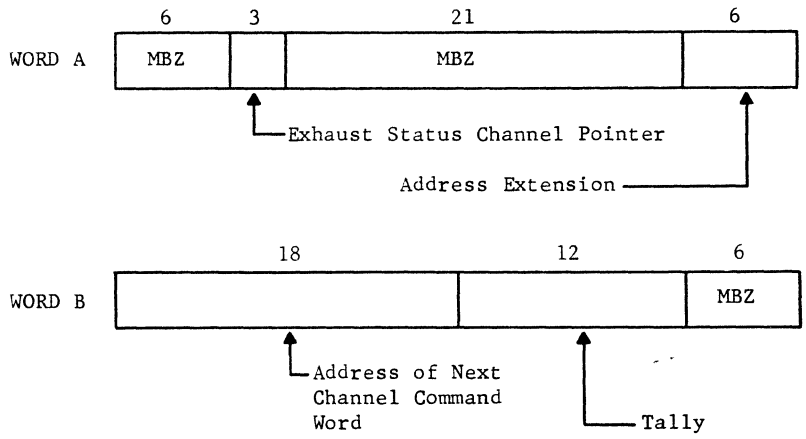
Connect Operand Word

The connect operand word initiates a GIOC action by specifying whether the operation is an operational or diagnostic activity and which of the three connect channels to use. The connect operand word is referenced by the Connect command issued by the software. The word format is:



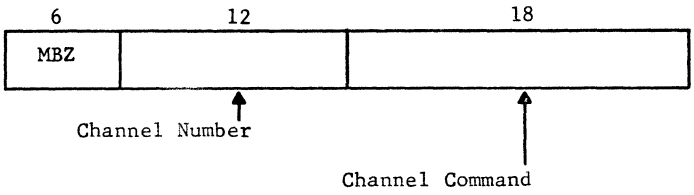
Command Pointer Word (CPW)

The CPW points to a list of channel command words for the GIOC. Each time service is granted to a connect channel, the GIOC uses the CPW to obtain the next channel command word (CCW) and send it to the channel specified in the CCW. The GIOC then updates the CPW by incrementing the address field and decrementing the tally field. When the tally field becomes zero, a status word and program interrupt notifies the software of the condition. A CPW is located in each of the three connect channel mailboxes. When the tally field decrements to zero (indicating the CCW list has been deleted) a status word is formed and stored using the exhaust status channel pointer field in word A. The formats of the two words are shown on the following page.



Channel Command Word (CCW)

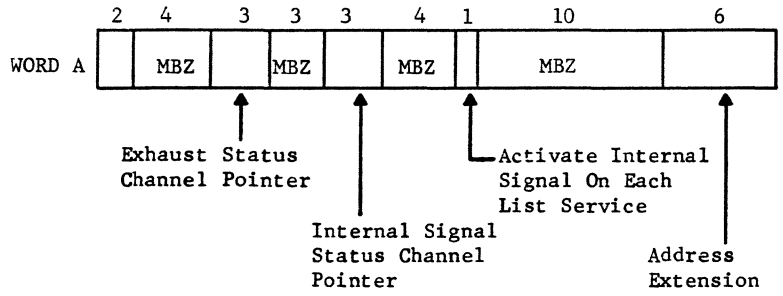
The CCW is the word referenced by the command pointer word. The channel number field indicates the specific channel that is to respond to the command. The channel command field contains the command itself. The format for the CCW is:

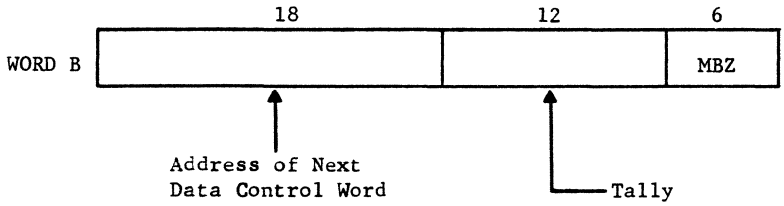


List Pointer Word (LPW)

A list channel with a list pointer word in a mailbox is associated with each data channel in the system. When a list channel service request is granted, the GIOC uses the LPW to obtain the next DCW from a DCW list in memory. It then transfers it to the mailbox location for the data channel (if the adapter is indirect) or to the data channel of the adapter (if the adapter is direct).

Word B contains the address of the list of DCW's and a tally field indicating the number of DCW's remaining in the list. When the tally field decrements to zero (indicating the DCW list has been depleted), a status word is formed and stored using the exhaust status channel pointer field in word A. A program interrupt is initiated following storage of the status word. The formats for the two words are shown on the following page.





Data Control Word (DCW)

Five basic types of DCW's are used by the GIOC:

1. Microcode DCW
2. Control Character DCW
3. Tally Match DCW

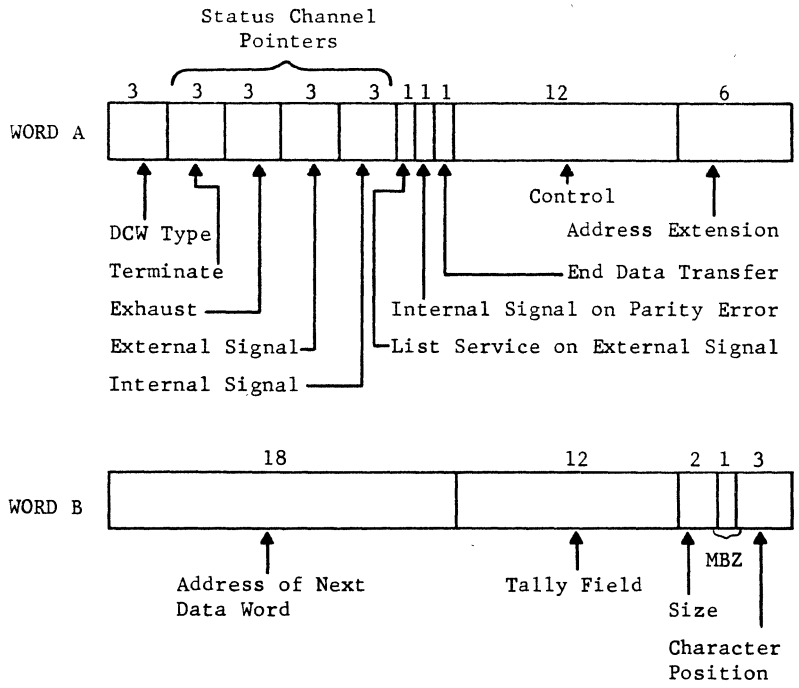
} Control the normal flow of data between memory and the adapters.

4. Command DCW
5. Literal DCW

} Transfer a portion of the actual DCW to the adapter as a command, or a literal character to be transmitted.

A DCW is located in the mailbox directly following the LPW for each data channel of each indirect adapter. The DCW is accessed each time the data channel requests service. DCW's for direct adapters are not in mailboxes, but are stored in the data channel of the adapter. The DCW's in a DCW list in contiguous memory locations. Normal termination of an I/O operation occurs when all of the DCW's in a list have been used.

GENERAL FORMAT. Each of the five types of DCW's conform to a general format. The general format is described in detail, and the manner in which each of the five types of DCW's differs from the general format is described. The general formats for the DCW words are shown on the following page.



Word A fields of the general format of the DCW are described as follows:

DCW TYPE. This field defines the DCW type.

STATUS CHANNEL POINTERS. These four fields point to the status channels to be used by the GIOC in forming and storing status words when various types of status events occur. The software may place a zero in any of the pointer fields if it does not want a status word stored at the occurrence of a particular event. The software cannot cause normal status events to make use of status channel zero, which is the emergency status channel. The four fields used to indicate the handling of four types of status events are:

Terminate Field. A terminate event indicates that program action is required before any further data transfer on that channel can take place. Terminate occurs only when a terminate signal is received from the adapter. The adapter sends the terminate signal as a result of a terminate condition occurring in the adapter or in response to an end-data-transfer signal from the controller.

Exhaust Field. An exhaust event indicates that the current operation is to continue under control of a new DCW which the GIOC obtains via the list pointer word before the next data transfer occurs. The activate-list-channel signal is sent to the adapter (at the same time exhaust occurs) to initiate the process of obtaining the new DCW. Exhaust occurs when the end-data-transfer field of word A is zero and one of two events takes place:

1. The tally field of the DCW (word B) goes to zero.
2. An artificial exhaust is created by a control character DCW.

External Signal. This indicates that a significant status change was detected in the adapter other than that which would cause a terminate. An example of such an event is the receipt of a special interrupt signal from a peripheral or a ring indicator from a communication terminal.

Internal Signal. This indicates the detection of a significant event in the GIOC controller, for example, a character match or detection of a parity error.

LIST SERVICE ON EXTERNAL SIGNAL. With a 1 bit in this field, the adapter indicating an external signal, and no termination signal present, the controller causes the adapter to activate the list channel associated with the data channel currently being serviced. The status word generated as a result of the external signal is stored normally.

INTERNAL SIGNAL ON PARITY ERROR. With a 1 bit in this field, the controller produces an internal signal immediately upon detection of a parity error in data from the adapter. When a 0 bit is in this field and a parity error is detected by the controller, the controller sets word B, position 32, to the value of 1 to remember the occurrence of the parity error.

END-DATA-TRANSFER. With a 1 bit in this field, an end data transfer signal is sent from the GIOC to the channel when the DCW list is exhausted. This signal usually causes communication channels to terminate and peripheral channels to either terminate or activate the list channel.

CONTROL. See the individual DCW types for explanation of this field.

ADDRESS EXTENSION. This field supplies the high-order 6 bits of a 24-bit memory address specifying the memory location for data transfers.

Word B fields of the general format of the DCW are described as follows:

DATA ADDRESS. This field supplies the low-order 18 bits of a 24-bit memory address specifying the location to be used by the GIOC for data transfers. This field is incremented by the GIOC as successive words are transferred.

TALLY FIELD. This field specifies the amount of data to be transferred between memory and the data channel. The field is decremented by the GIOC as successive words or characters are transferred. See exhaust status channel pointer description for word A for tally exhaust operation.

SIZE. This field specifies the size of the bytes to be transferred between memory and the adapter. These may be 6-bit characters, 9-bit characters, or 36-bit words. This field also specifies whether the tally is a character count or a word count.

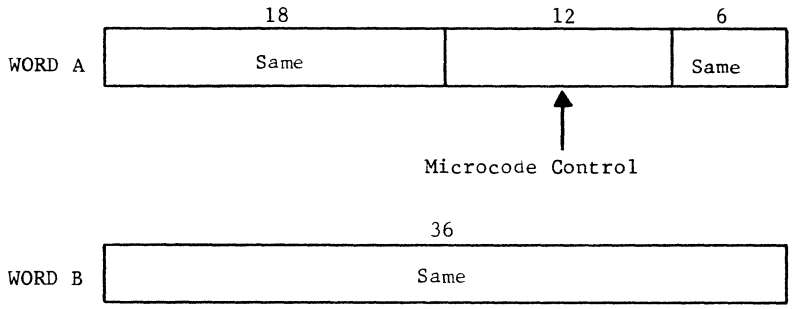
BIT 32. This bit position must be set to zero initially by the software. It is possible that this position contains a one as a result of bit position 16 having been set to zero to allow bit position 32 to go to a one to remember a parity error. In this case, the controller produces an internal signal when a terminate or exhaust occurs.

CHARACTER POSITION. This field specifies which character position in the current data word is to be used for the next character transfer.

The following descriptions of the five types of DCW's point out how each differs from the basic DCW just described.

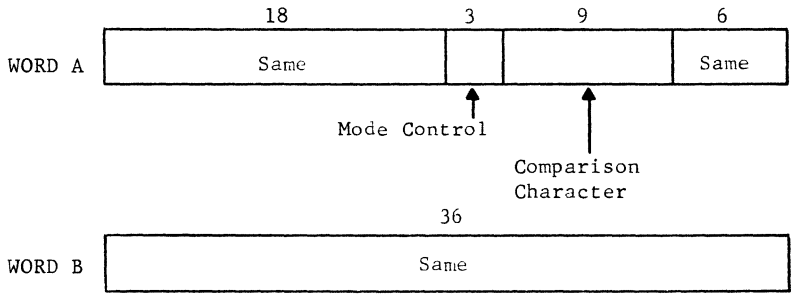
Microcode DCW

The microcode DCW has the general format. When it is used for normal data transfers, it requires that the microbits be zero. The microcode bits can be used to detect ASCII control characters or to inhibit access to memory until after the current DCW exhausts. The formats of the two words are:



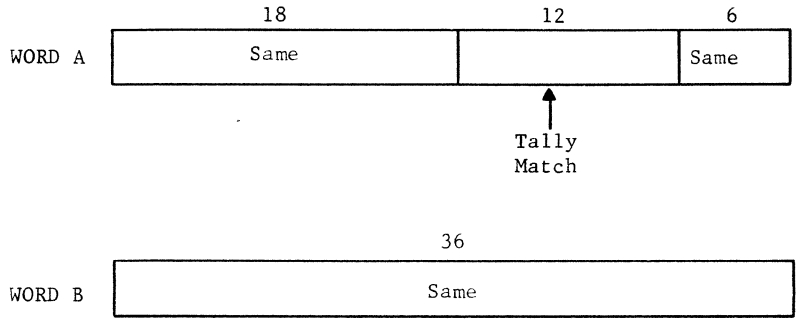
Control Character DCW

The control character DCW differs from the basic DCW only in the mode control and comparison character fields. The mode control field indicates the type of status word to be stored if the received character matches the comparison character field. The size field in word B determines the size of the characters to be compared. Character comparison occurs only if the size field indicates that 6- or 9-bit characters are being transferred. The comparison character field is compared to each incoming character as it is stored in memory. The formats for the two words are:



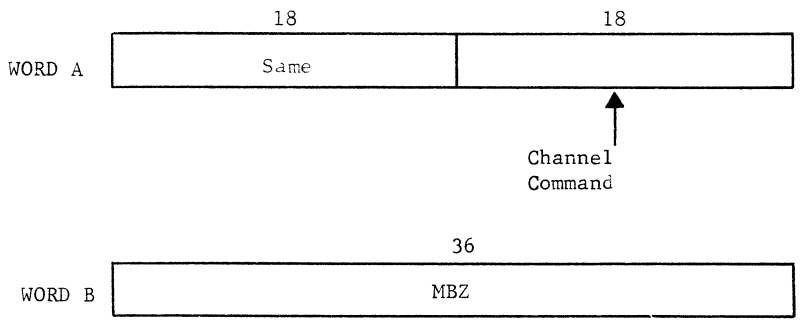
Tally Match DCW

The **tally** match DCW differs from the basic DCW only in the tally match field. This field notifies the software, via a status word and interrupt, when a certain tally exists as data is transferred between memory and an adapter. When the tally field is equal to the tally match field at the time the DCW is accessed, an internal signal is generated. This results in the formation and storage of a status word via the internal signal status channel pointer and an interrupt. The formats for the two words are:



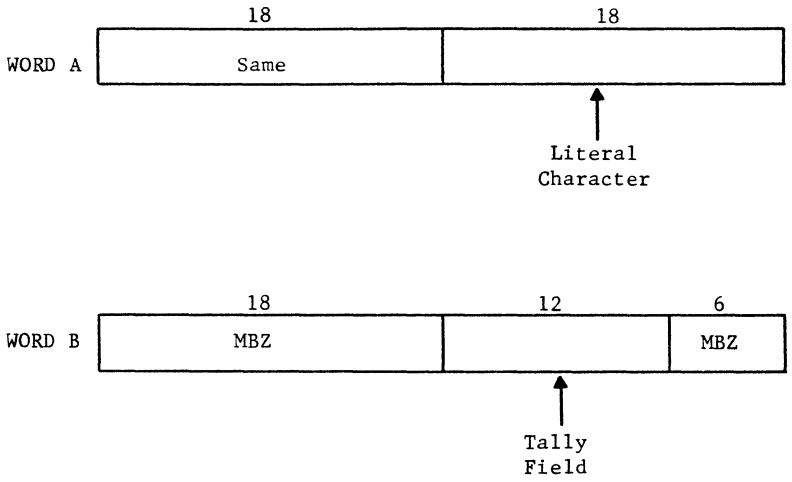
Command DCW

The command DCW issues commands to data channels at certain preplanned points in the data transfer sequence. This DCW is placed within a list of DCW's for a particular channel and is accessed by a list pointer word during list channel service. This DCW is sent directly to the adapter and is also placed in the memory mailbox so that the status channel pointers are available for subsequent use. The coding of the channel command field of word A is similar to that of the channel command word described on page 59. The formats for the two words are:



Literal DCW

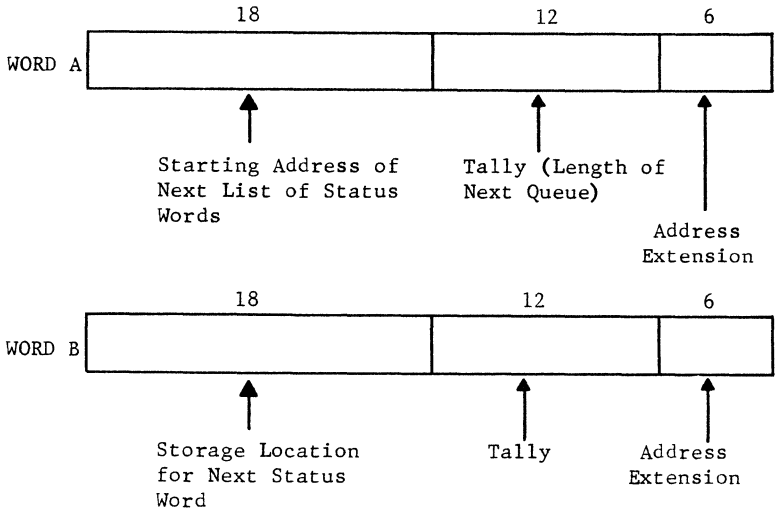
The literal character field of word A specifies a literal character, including start stop bits if necessary, to be transmitted by a communication channel. The tally field of word B determines the number of times the character is to be transmitted. The formats for the two words are:



This DCW is useful in such applications as sending fixed-length pauses, marking or spacing a line, and sending synchronization characters.

Status Control Word (SCW)

An SCW is located in each of the first eight mailboxes to control the transfer of status words from the eight status channels to memory. With the exception of status channel zero (the emergency status channel) the status channels may be assigned priority any place in the priority structure. The formats for the two words are:



Each time the GIOC stores a status word in memory, the address field of word B of the appropriate SCW is incremented and the tally field decremented. When the tally field reaches zero, the contents of word A (if non-zero), replace the contents of word B, and word A is set to zero. A program interrupt will be initiated following the storage of each status word.

STATUS WORDS

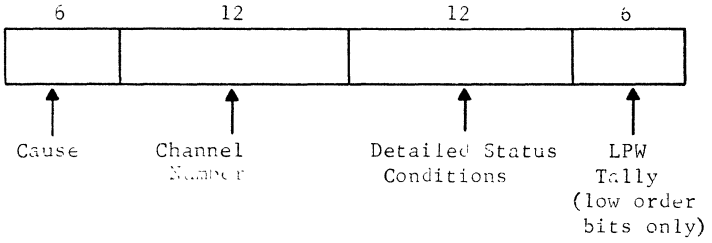
Three types of status words are generated and stored by the GIOC. Two types of status words result from noncatastrophic status changes occurring in the various adapters and their associated peripherals or communication lines. These status events are events such as parity errors in data received, data not transferred rapidly enough for the GIOC to keep up with the input/output device, detection of a specified character in the input, or the exhaustion of a specified DCW. Type A or B status words are stored when noncatastrophic status changes occur. (See status word formats which follow). Type A is stored for all terminate or exhaust events. Type B is stored as a result of an external or internal signal. The third type of status word is the emergency status word. It is stored to indicate the apparent failure of a memory module or failure of the software.

When simultaneous events cause two or more status channel pointers to indicate the same status channel, the status word format used is according to the following priority:

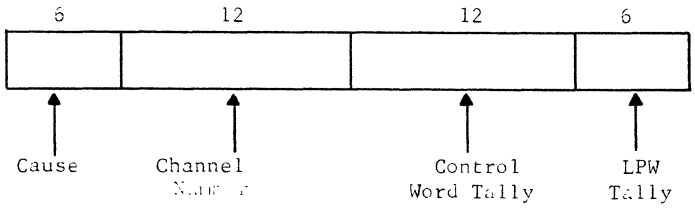
Highest	Terminate
↓	Exhaust
	External Signal
Lowest	Internal Signal

Following are the formats for the three types of status words:

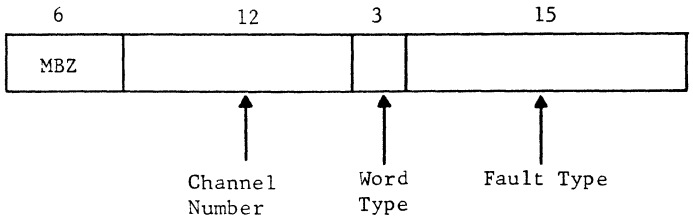
Type A — This status word is stored for all terminate or exhaust events. The word shows all the status conditions.



Type B — If an external or internal signal occurs, a Type B status word is stored showing the new tally residue and four of the status conditions.



Emergency Status Word — This status word is stored to indicate the apparent failure of a memory module or of the software.



PRIORITY

Every function performed by the GIOC is assigned a priority. In other input/output systems, when a condition is detected which requires extra time during a normal service cycle, the time is taken immediately, regardless of the priority of the data channel involved. This may cause a low-priority function to keep control of the system while a higher-priority data channel waits. In the GIOC, however, the detected condition is retained by a priority network which guarantees its handling before the applicable data channel is serviced again. The GIOC then proceeds to take care of higher priority functions involving other channels. The maximum system can have 96 levels of priorities: 81 for adapters and 15 for controller channels. All adapter service request lines pass through a patch plug where priorities are assigned to the requests. Service requests of the connect, status, and diagnostic channels also pass through the patch plug for priority assignment.

FAULTS AND INTERRUPTS

Fault Detection

The GIOC can detect two kinds of faults: memory faults and control word faults. At the end of each memory cycle, the memory module indicates to the GIOC whether or not an illegal act (memory fault) occurred during that cycle. The GIOC then gives notice to the program or operator.

Control word faults occur when either the hardware or software has errors which prevent handling of control words (CPW,LPW,DCW, or SCW). When a control word fault occurs, the GIOC does not update the bad control word. The faulty control word remains in its mailbox until the software causes it to be modified.

In handling faults, the controller can take the following types of action:

- Notify the software by placing a status word in one of the normal status channels. Cause the applicable adapter channel to terminate data transfer. This indicates a user-program error but no malfunction of hardware or software.
- Halt and sound an audible alarm to indicate a breakdown in the communication path to the software. Lights on the control panel enable the field engineer to determine the cause of the halt.

Interrupts

Interrupts occur when the GIOC stores a status word in memory to inform the processor about a condition. Each of the eight GIOC status channels is assigned to one of the 32 interrupt cells in the memory module. Each time a status word is stored by the GIOC, the GIOC controller issues a Set Execute Interrupt Cells (SXC) command to the memory which has the GIOC base address. The memory module then notifies the software that at least one status word has been stored by the status channel which corresponds to the interrupt cell which was set. This causes the processor to execute the routine which services the list of status words associated with the interrupt.

8. MEMORY MODULE

The memory modules in a system have three functions. Core memory stores programs and data. Paths allow processors, GIOCs, and drums to send control information to each other. Clocks provide date and time-of-day information and have the ability to interrupt the system at a specific time.

ORGANIZATION

A memory module has two parts: a controller and core memory. The controller receives, interprets, and executes the commands sent to the memory module by other modules in the system. Programs and data are stored in core memory. The control information paths, the clock subsystem, and storage for these paths and the clock are part of the controller.

Storage Function

A memory module may contain 32k, 64k, or 128k of 36-bit words (plus parity), and has a cycle time of one microsecond. Either one or two words may be read or stored in one memory access. A zone control feature may also be used to store some characters in a word without disturbing the other characters in that word. Either 6-bit or 9-bit characters may be stored in this way.

In order to store or retrieve information, a processor, GIOC, or drum sends a command and the necessary data to a memory module. The memory module executes the command and either stores the received data or sends the desired data to the requesting module. Any conversion from relative to absolute addresses is done by the requesting module.

A typical GE-645 system has more than one core memory module. Memory access requests are distributed among the physical memories by a memory interlace technique. This equalizes the load among the physical modules and increases system performance by decreasing the competition and queuing of requests for the same physical memory. The circuitry that controls interlacing is located in the processors, GIOCs, and drums; but interlacing is discussed in this chapter because of its close connection with memory modules.

There can be two-way interlace, four-way interlace, or no interlace. When interlace is not used, sequential memory addresses are located in the same physical memory, neglecting the fact that an individual memory is of finite size. Memory addressing is usually done by pairs of words, so it is more correct to speak of pairs of memory addresses. Therefore, with four physical memories, A, B, C, and D, and no interlace, sequential pairs of addresses fall in succession into a single memory until that memory is filled, then start filling one of the other memories. Two-way interlace has sequential pairs of addresses stored so that the first pair is in memory A, the second pair in B, and the third pair in A etc. With four-way interlace, the sequential pairs of

addresses rotate among the four memories--A, B, C, D, A, B, C, D, etc. The effect of the types of interlace is summarized in Table 3. Memory A is arbitrarily selected as the initial memory.

TABLE 3
TYPES OF INTERLACE

Word Pair	Type of Interlace		
	None	2-Way	4-Way
START+ 0, 1	Memory A	Memory A	Memory A
START+ 2, 3	A	B	B
START+ 4, 5	A	A	C
START+ 6, 7	A	B	D
START+ 8, 9	A	A	A
START+ 10, 11	A	B	B
START+ 12, 13	A	A	C
START+ 14, 15	A	B	D

Control Information Paths

When a memory module receives a Connect command from a processor, the memory controller forwards a connect signal to the specified module. The connect signal is accompanied by a word of control information for the recipient module.

Interrupt cells notify Multics-645/I that some event has occurred which needs attention. A group of 32 interrupt cells are located in the memory controller. Each cell is one bit of storage. Cells are set by a clock subsystem or by a command received from a processor, GIOC, or drum. When an interrupt cell is set, the controller notifies a processor of this fact, along with the cell number. The processor stops what it was doing and transfers to the Multics-645/I routine for that particular interrupt cell. Multics-645/I can temporarily inhibit the interrupting action of any interrupt cell or cells. This is done with the aid of a 32-bit interrupt mask register that is also located in the controller.

CLOCKS

There are two types of clocks: a calendar clock and an alarm clock. The calendar clock provides the current date and time. Multics-645/I sets the alarm clock for a certain time. The system is then interrupted when that time is reached.

The calendar clock is a 52-bit register that is changed at one microsecond intervals. This provides a capacity greater than one century without overflowing, and a precision of one microsecond. The necessary accuracy and reliability is obtained with special circuitry. The alarm clock has the same greater than one century capacity and a resolution of 64 microseconds.

On rare occasions when it is necessary, the calendar clock will be set by a General Electric field engineer. Since Multics-645/I operates on Universal Time, the field engineer sets the calendar clock to the number of microseconds since midnight, January 1, 1901, Greenwich Mean Time. A processor may read the calendar clock at any time. Provision for setting the calendar clock

under program control is deliberately omitted. Therefore, programming errors or hardware malfunctions outside the clock cannot destroy the current time. The alarm clock is set to a particular value by Multics-645/I. Then, the alarm clock's time is continuously compared to the changing time in the calendar clock. The comparison is done by special hardware in the clock subsystem. When the calendar clock reaches the time preset in the alarm clock, an interrupt cell is set. This notifies Multics-645/I through a program interrupt.

Because of the importance of the clocks to a system, a backup source of power may be provided. Should the primary power fail, the clock subsystem automatically switches to the backup source and sets an interrupt cell. Multics-645/I then notifies the operating personnel.

9. DRUM MODULE

The High Performance Magnetic Drum (MDU302) is used as an extension of memory. Segments and pages of programs flow between memory and a drum under the control of the operating system. Information used frequently remains in core memory, and information needed less often is on the drum where it can be obtained quickly when needed.

ORGANIZATION

A drum module consists of two units: a controller and a rotating storage unit. The controller obtains control words from memory, interprets them, and reads or writes the desired information from or onto the rotating unit.

Data is organized in sectors of 80 words, 64 of which are data words and 16 are in the guard band. Words in the guard band are used to store parity for the 64 data words and for testing drum operation without disturbing the data words. (This testing feature is discussed later in this chapter.) A track set of 16 read/write heads simultaneously reads or writes a sector.

Drum modules are available in several sizes. Storage capacity ranges from 1 to 16 million words. A typical drum size is 4 million words organized into 4096 tracks, and therefore, 4096 read/write heads.

PERFORMANCE CHARACTERISTICS

The transfer rate between a drum and memory is 470,000 words per second. Data is always transferred as pairs of words, that is, 72 bits per memory request, with four words transferred every 6.7 microseconds. At this rate, 32k words are transferred in 55 milliseconds. Since there is a guard band at the end of each sector, there will be a small amount of time between the end of one sector and the beginning of the next sector. Figure 32 represents a small portion of a drum's surface. Each rectangle represent an 80-word sector, which includes a guard band.

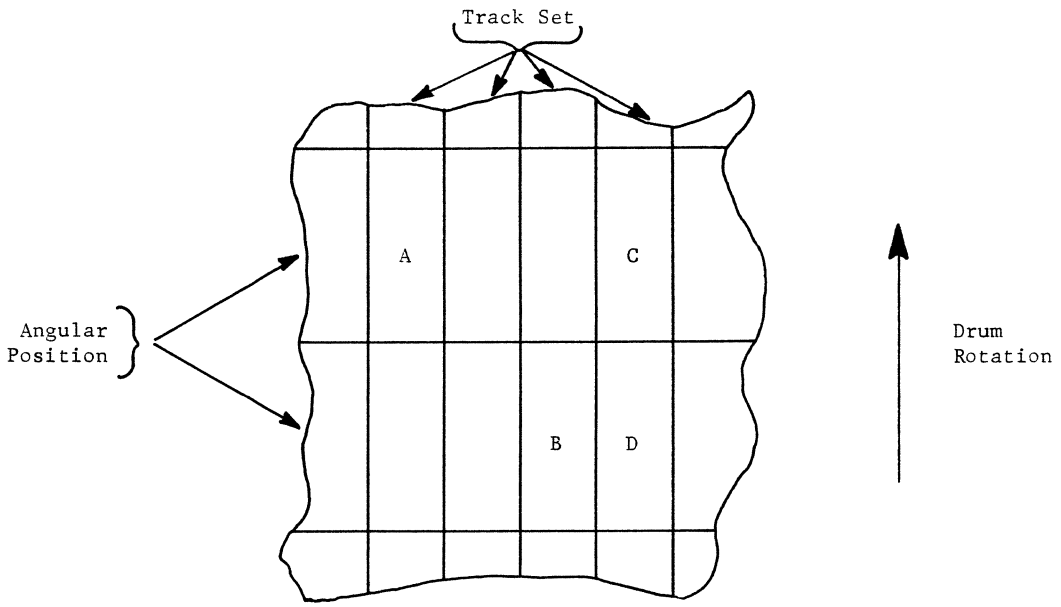


Figure 32. Successive Sector Capability

Assume that sector A is being written. After writing the last of the data information, new parity is recorded in the guard band, but the other guard band information is not changed. The drum can obtain a new control word from memory, interpret that word, select a different set of read/write heads, and change from the writing to the reading mode before the beginning of sector B comes under its set of read/write heads. Thus, it is possible to eliminate the drum's latency by building the list of control words in the proper order. Sectors can be identifed by their angular position and by the track set used in reading and writing. In Figure 32, A and C start at the same angular position, and B and D start in the next angular position. C and D are read or written by the same track set.

OPERATION

The drum differs from the GIOC in that, under normal circumstances, it never terminates its operation and hence does not have to be repeatedly connected. A list of data control words (DCW's) is continually updated by the operating system. Each DCW specifies a drum command, a memory area, a drum area, and the location of the next DCW. The drum area may be either one 64-word sector or sixteen consecutive sectors with a total of 1024 words. These drum areas correspond to the page sizes used by a processor.

The drum portion of the operating system receives requests from other parts of the operating system, determines what drum sectors are involved, makes up the DCW's and inserts the DCW's in a list. The DCW specifies a drum area by track set, angular position, and number of sectors. The drum latency is minimized by the order in which the DCW's are placed in the list.

Instead of the order corresponding to the order in which the requests are received by the operating system, it is determined by the angular position of the drum areas. Thus, when the drum reaches a new angular position, the next DCW is for a sector at that position.

At times, the drum is ready to service the list of DCW's before requests have been received for sectors in every angular position. In this situation, the operating system uses a DCW which specifies that no data transfer is to take place. This allows the drum to maintain continuous operation by eliminating a disconnect and then a connect.

A current status word (CSW) is stored in memory as the drum begins "executing" a DCW. If an error of any kind is detected by the drum, an abnormal status word (ASW) is stored in core memory. The contents of the ASW identify the error. In addition to storing an ASW, the drum initiates a program interrupt by setting an interrupt cell in a memory module.

Abnormal status words are stored in a queue of 32 words. When the end of the queue is reached, the drum automatically starts over at the beginning of the queue's storage area. Hardware is provided in the drum so that the operating system can indicate which ASWs it has serviced and, hence, which ASW locations the drum may use for storing new ASW's. If the drum reaches a point where it would store a word on top of a word the operating system has not yet used, it disconnects and signals this event by a program interrupt.

The drum also can store an ASW and cause an interrupt even though no error is detected. This is done with specific DCW commands, such as "read and interrupt." Thus, when that DCW is used, the operating system knows where the drum is in the list of DCWs. This may be used to warn the operating system that the drum is near the end of the DCW list and that more DCWs will be needed soon.

When a program interrupt is needed, the drum indicates the relative seriousness of the interrupt by setting one of three possible interrupt cells. The actual cells used are determined by a patchboard in the drum. Normally, the programmed interrupt cell is set to indicate that the selected DCW has been used. A different cell is set when a data error is detected. The third cell is set if control problems are detected. In all three cases, the accompanying ASW contains bits which further define the reason for the interrupt.

TEST MODES

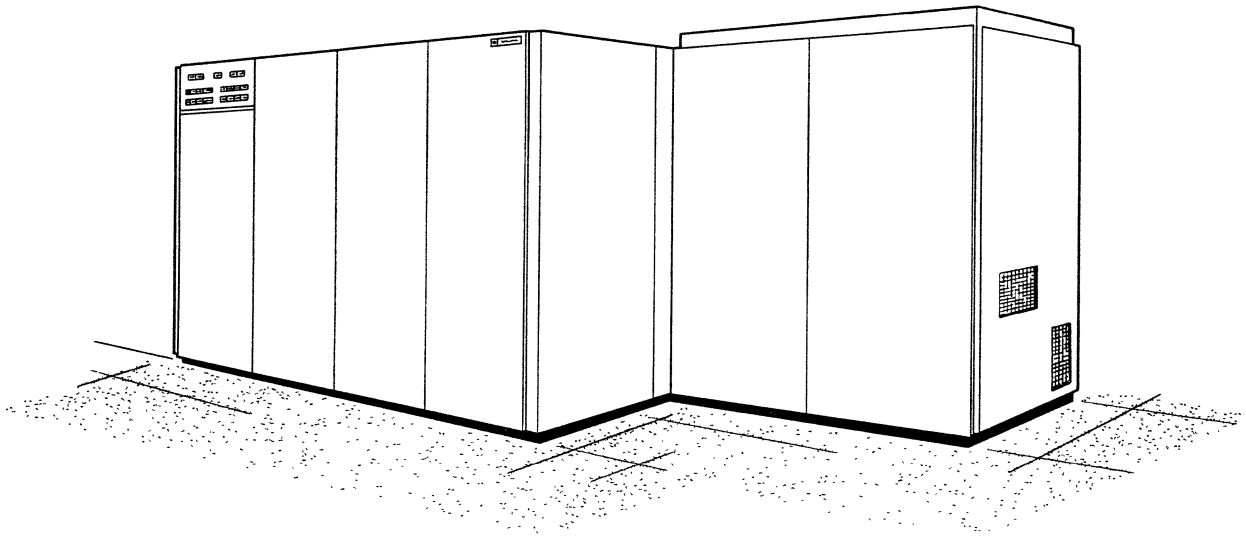
Special test modes enable extensive test and diagnostic programs to be run without disturbing the user's data. There is a special test sector of eight 36-bit words within the guard band of each sector. When testing, data can be transferred between the test sector and core memory. Every addressable sector recorded on the drum can be accessed, and associated logic and recording electronics can be checked without altering or disturbing user data recorded on the drum.

10. PERIPHERAL AND TERMINAL EQUIPMENT

The common peripheral and remote terminal equipment described in this section communicate with the GE-645 system through the GIOC. A summary is given of the characteristics of each type of equipment. The following are described:

Disc Storage Unit (DSU250) and Controller (DSC250)
Removable Disc Storage Unit (DSU150) and Controller (DSC150)
Mass Storage Unit (MSU388) and Controller (MSC388)
GE-115 Information Processing System
DATANET-760 Display Terminal Unit (DTU760)
Magnetic Tape Subsystem (Controllers MTC401 and MTC406 and numerous Tape Units)
Card Reader and Control (CRZ201)
Perforated Tape Subsystem (PTS200)
Card Punch and Control (CPZ201)
Printer and Control, On Line, 136 Column, 1200 lpm (PRT201)
ASCII Extended Character Set Printer (PRT202)
Peripheral Switch Console (PSC200)
Programmable Peripheral Switch (PS6010)

DISC STORAGE UNIT (DSU250)
DISC STORAGE CONTROLLER (DSC250)



DSC250 DSU250

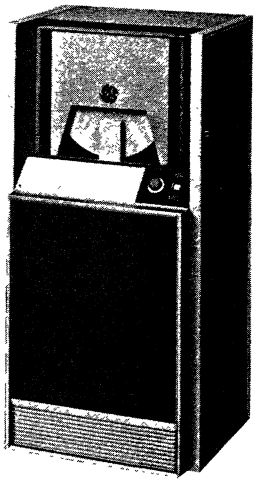
CONFIGURATION	1 controller (1 or 2 channels-simultaneous operation). 1,2,3, or 4 file units. 16, 24, or 32 discs per unit. A maximum system (2 controllers having 2 channels each, and 4 file units) has 4 independent access paths to any disc unit.
DATA MEDIUM	31-inch discs.
SPEED	300,000 character/second transfer rate per channel. Simultaneous read or write on 1, 2, 3, or 4 channels. Maximum transfer rate with 4 channels, and 4 file units is 1.2 million characters/second.
OPERATIONAL MODES	On-line reading or writing.
CHECKING	Parity check on data transmitted to the controller unit. Parity check on data transmitted between controller and file units. Modulo-2 parity check on each character received. A validity check on arm position. Compare and Verify command for bit-by-bit comparison of data recorded versus data sent by the computer. Transmission timing checks. Sector address confirmation on all sectors read or written. Timing check by clock pulses.

CAPACITY		
	One 16-disc unit	100 million 6-bit characters
	One 32-disc unit	200 million 6-bit characters
	Four 32-disc units	800 million 6-bit characters

FEATURES

A single controller can control up to 800 million characters.
16 simultaneous accesses on one file unit.
64 simultaneous accesses on a maximum subsystem.
Up to 1024 character records may be read or written from one placement of any one positioning arm.
Eight read/write heads service each disc surface. Each read/write head services 64 data tracks.
Modularity permits adding storage in 50 million character increments above the initial 100 million character base.

REMOVABLE DISC STORAGE UNIT (DSU150)
DISC STORAGE CONTROLLER (DSC150)



DSU150

CONFIGURATION	1 Controller (1 or 2 channels - not simultaneous). 1 to 8 disc units (each in separate cabinet). Each disc unit accepts one removable cartridge containing one disc.
DATA MEDIUM	16-inch removable discs.
DATA FORMATS	Each disc has 40,960 addressable sectors of 192 characters each. A total of 512 sectors (98,304 characters) are accessible in each actuator position.
SPEED	Transfer rate 281,000 6-bit characters per second. Average positioning time 95 milliseconds, average latency time 25 milliseconds.
OPERATIONAL MODES	On-line reading or writing.
CHECKING	Parity check on data transmission from the user. Parity generated on data transmission to the user. Check on command sequence, device codes, operation codes, and select data. A 12-bit checkword generated on each 192 character data block recorded on a file unit. Compare and verify command for bit by bit comparison of user data versus recorded data. Transmission timing checks. Data block address confirmation on all sectors read or written.

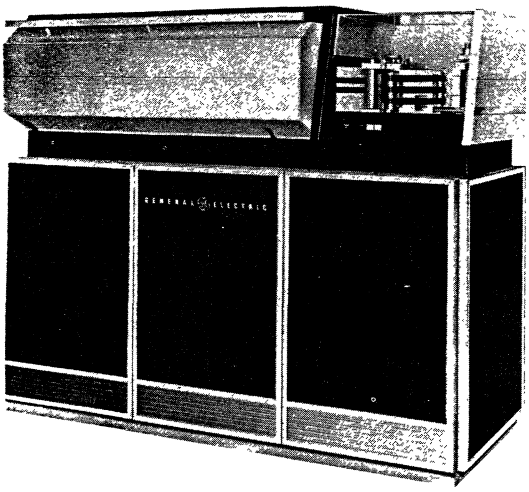
CAPACITY

1 removable disc	7,864,320 6-bit characters
1 disc drive unit	7,864,320 6-bit characters
8 disc units	62,914,560 6-bit characters

FEATURES

Discs are mounted vertically and are easy to change.
They are lightweight and easy to handle.
Recording surfaces are kept dust free because access to surfaces is possible only when the unit is operating.
There can be eight simultaneous seeks per subsystem.

MASS STORAGE UNIT (MSU388)
CONTROLLER (MSC388)
ADDITIONAL MAGAZINE STORAGE (AMS388)



MSU388

CONFIGURATION	A subsystem consists of a controller and from one to four mass storage units. Each unit can have additional magazine storage. A storage unit has 8 removable magazines of 256 magnetic cards and can have additional storage of 8 more magazines.
DATA MEDIUM	Data is recorded on one side of 16" by 4-1/2" flexible magnetic cards.
DATA FORMAT	Information is recorded serially on 128 tracks extending the length of each card. Tracks are paired into 64 bands, each divided into 4 longitudinal blocks. A block stores 650 characters, (6 bits plus parity). A card has 166,400 characters (256 blocks).
SPEED	Maximum transfer rate is 80,000 characters per second.
OPERATIONAL MODES	On-line reading or writing.
CHECKING	Automatic read after write parity check. Full address verifications before card entry onto the drum. Parity check during reads, including correction of every single bit error.

CAPACITY

Block	650 characters
Card	166,400 characters
Magazine	42,598,400 characters
8-Magazine Unit	340,787,200 characters
16-Magazine Unit	681,574,400 characters
4-Unit Subsystem	2,726,297,600 characters

FEATURES

Simultaneous select and preselect operations on all units, therefore four simultaneous select operations on subsystems with four units.

A storage unit can read or write up to 256 blocks on one card by a single instruction.

GE-115 INFORMATION PROCESSING SYSTEM

The GE-115 is ideal as a remote input/output terminal connected by communication lines to the GE-645 system. It can be taken off line and will perform as a card processing system. As a remote terminal to a larger computer, it provides direct access to the larger system without the media conversion often required at peripheral processors.

CENTRAL PROCESSOR

Memory Unit

Cycle time: 8.0 microseconds.
Data format: 8-bit characters or octets, plus parity.
Sizes available: 4096 and 8192 words.
Storage: magnetic core.
Instruction format: variable.
Addressing mode: direct character addressing in binary.
Operation mode: asynchronous.
Checking: parity checking for each octet.
Instructions: a total of 28 primary arithmetic, logical, editing, and transcoding instructions. AND, OR and EXCLUSIVE OR enables program to modify or test memory bits.

Control Unit

This unit fetches and interprets instructions from memory. It establishes connections with input/output units specified in the instruction. The control panel permits manual intervention to guide system operation.

Arithmetic Unit

Performs both decimal and binary arithmetic operations. Decimal operations can be performed on 16 digit numbers. Binary arithmetic instructions are performed on fields up to 16 octets.

Maximum words transfer speed is 125 kw/second.

GE-115 AS A CARD SYSTEM

With a choice of a card reader, two card punches, and a line printer, the system can operate off line from the GE-645 and be tailored to fit most punch card center requirements.

Peripheral units for the card system are:

Card Reader (CRZ100)

Speed: 300 cards per minute (standard 80-column, decimal code).
Reading method: serial.
Hopper capacity: 500 cards.
Stacker capacity: 500 cards.

Card Punch (CPZ101)

Speed: up to 200 cards per minute.
Punching method: serial.
Hopper capacity: 1500 cards.
Stacker capacity: 1500 cards.

This unit can punch 80 columns in each card at 60 cards per minute. Speed can increase to 200 cards per minute for partially punched cards.

Card Punch (CPZ103)

Speed: 300 cards per minute.
Punching method: parallel.
Hopper capacity: 1200 cards.
Stacker capacity: first stacker - 1200 cards.
second stacker - 100 cards.

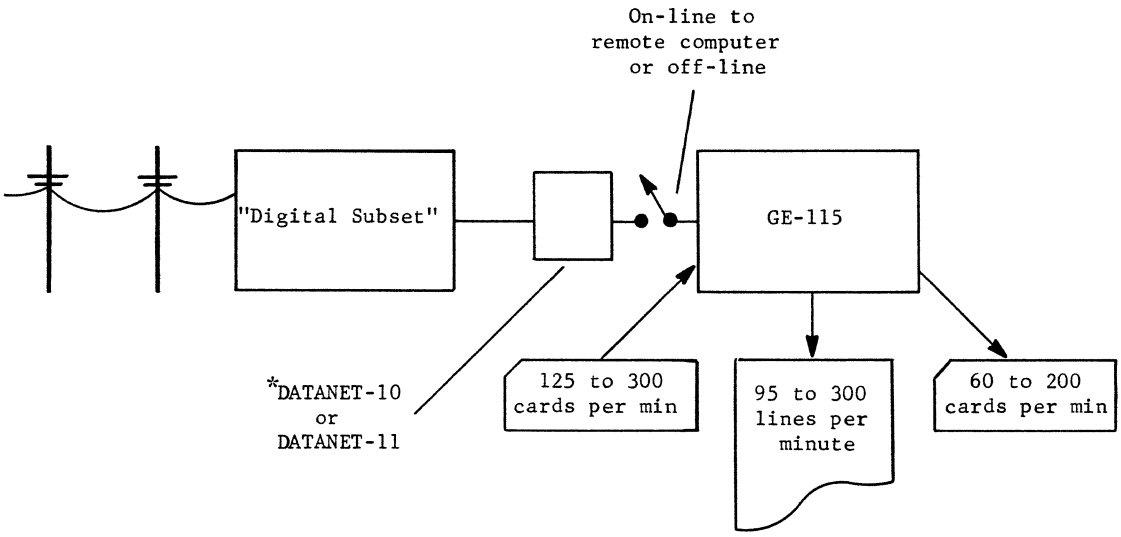
Line Printer (PRT100)

Speed: 300 lines per minute.
Positions: 136.
Characters: 64 - standard General Electric print character set (ASCII).
Spacing: horizontal - 10 characters per inch vertical - 6 lines per inch.
Skipping: 12 inches per second.
Form width: 3 to 22 inches.
Operation mode: asynchronous.

GE-115 AS A COMMUNICATIONS TERMINAL

The GE-115 is connected to the GIOC of the GE-645 system. Communication is via private or switched common carrier transmission facilities. The GE-115 can generate an interrupt at a distant computer, and read in a program request followed by data to be processed. When the larger computer has performed the requested tasks, it transmits the data to the GE-115 to be printed or punched into cards.

Using a data set 201A or 201B on a voice grade line, card-reading speed is 125 cards per minute minimum, card punching is 85 cards per minute minimum, or printing on a line printer is 95 lines per minute minimum. Printer speeds increase if printed lines are not a full 136 characters in length.



When data communication lines transmitting data at 2000 baud are used, the communications line interface (CLI100) is used with the data set 201A. The CLI100 is used for lines transmitting data at 2400 baud, with the data set 201B.

When used in scientific calculations, the GE-115 can perform the majority of the calculations and switch to the GE-645 only when it needs the speed and capacity of the larger system.

*Reg. Trademark of General Electric Company

DATANET-760 DISPLAY TERMINAL UNIT (DTU760)

The DATANET-760 is a remote terminal for computer processing. It provides convenient direct access, real-time entry of data into the computer system.

It displays incoming communication on a standard television monitor and transmits outgoing communication by use of a typewriter-like keyboard.

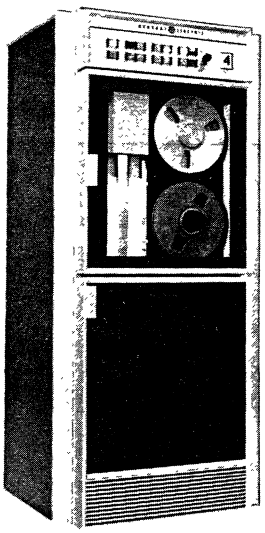
A printer can be connected to produce a hard-copy printout of material displayed. When hard copy is required, the operator selects the part of the information on the screen that is to be printed, and depresses the print button.

Data may be completely assembled and verified before it is sent to the computer. Changes, additions, or deletions can be made to existing records in the computer file by displaying the data on the screen and typing in changes on the keyboard. The cathode ray tube display has high brightness for use in normal office light.

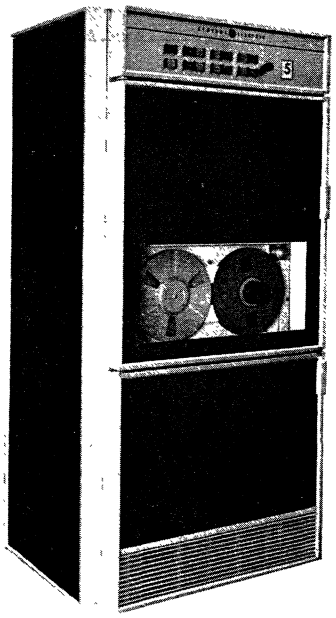
All operating controls (except those unique to the display monitor) are located on the keyboard unit.

An important feature of the DATANET-760 is automatic error recovery. If the computer receives a message containing an error, it causes the message to be retransmitted. If a predetermined number of retransmissions does not result in correction of the error, the computer sends a message to the sending terminal notifying the operator that manual recovery is necessary.

MAGNETIC TAPE SUBSYSTEM



MTH201,301,404,405



MTH211,311,411,412

A magnetic tape subsystem consists of one controller and one or more tape units. A GE-645 system may have more than one magnetic tape subsystem.

DATE MEDIUM	Half-inch-wide, magnetic-oxide plastic tape, up to 2400 feet long, 7- or 9-track (compatible with American Standard Association codes).
DATA FORMATS	Binary (standard) and special decimal.
CHECKING	Transfer timing Blank tape read Transmission parity Lateral parity Missing character Longitudinal parity Bit detected during erase
FEATURES	Tape handlers are available in either 7 or 9 tracks, and operate with controller MTC406 and MTC401.

CONTROLLER CHARACTERISTICS

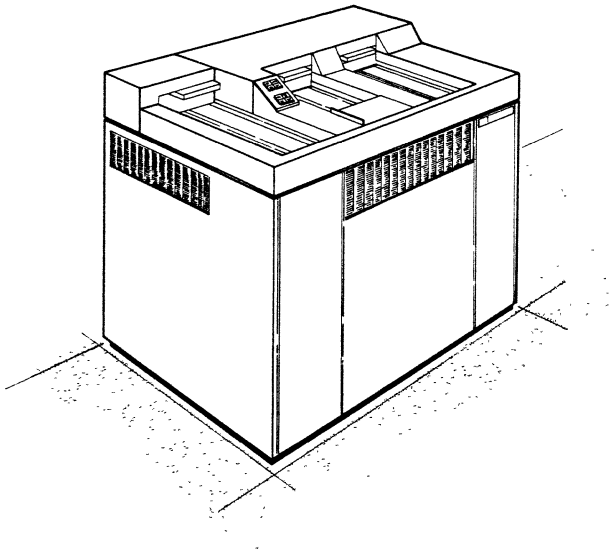
Controller Type No.	No. of I/O Channels	Maximum Number of Tape Units per Subsystem
MTC 406	2	16
MTC 401	1	8

Normally, the MTC 406 controller will have each of its two channels connected to a different GIOC.

TAPE UNIT CHARACTERISTICS

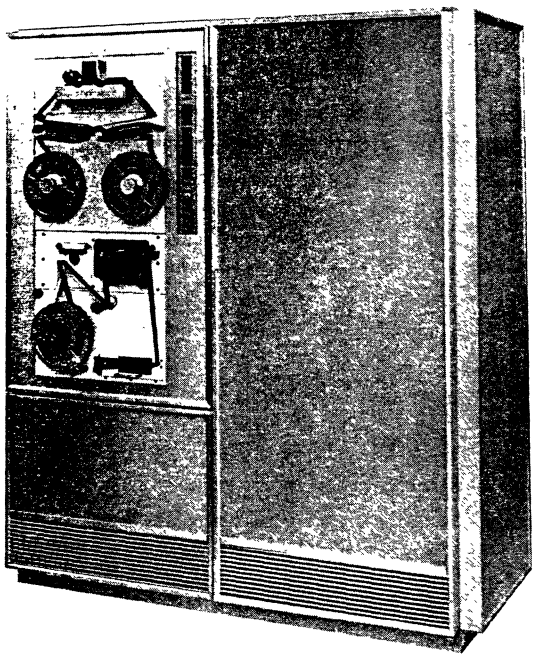
Tape Unit Type No.	No. of Channels	Tape Speed (ins. per sec.)		Recording Densities Available (bits per in.)	Data Transfer Rates for Various Densities (in thousands of characters per second)					
					8-bit characters			6-bit characters		
		Forwd.	Rewnd.		200	556	800	200	556	800
MTH412	9	150	300	200,556,800	30	83	120	40	111	160
MTH411	9	150	300	200,556	30	83		40	111	
MTH405	9	75	225	200,556,800	15	42	60	20	56	80
MTH404	9	75	225	200,556	15	42		20	56	
MTH311	7	150	300	200,556,800				30	83	120
MTH211	7	150	300	200,556				30	83	
MTH301	7	75	225	200,556,800				15	42	60
MTH201	7	75	225	200,556				15	42	

CARD READER AND CONTROL (CRZ201)



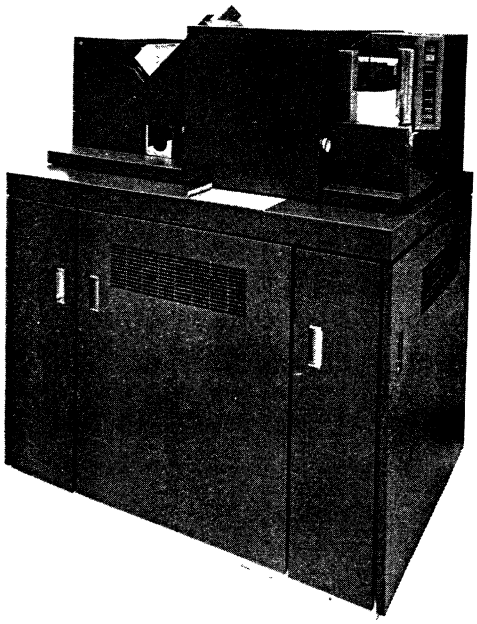
DATA MEDIUM	80- or 51-column cards with upper left or right corners cut round or square. Score-edge cards can be read.
DATA FORMATS	Standard decimal card code and column binary.
SPEED	900 cards per minute (80 columns). 1200 cards per minute (51 columns).
OPERATIONAL MODES	On-line decimal. Binary (memory image). Decimal and binary intermixed.
CHECKING	Card feed alert Card synchronization Read head alert Card jam Character validity (decimal mode) Hopper empty Stacker full
FEATURES	2000-card hopper and stacker capacity. Two output hoppers, hopper selected by the program. Dual read heads: data read at two independent stations and compared. Last batch control via LAST BATCH switch. 1000-card auxiliary hopper. Continued operation while loading or removing cards.

PERFORATED TAPE SUBSYSTEM (PTS200)



DATA MEDIUM	Paper or Mylar polyester film laminate tape perforated with chad-type holes.
DATA FORMATS	Reads and punches 5-, 6-, 7-, and 8-channel tapes, 10 characters per inch, in widths of 11/16, 7/8, and 1 inch; recognizes all possible code combinations.
SPEED	Reader, 500 characters per second; punch, 150 characters per second.
OPERATIONAL MODES	On-line: punching fully controlled by the GIOC (reading controlled by the GIOC and the plugboard in the perforated tape subsystem). Off-line: reading and punching controlled by the PTS200 Tape Reader/Punch control panel.
CHECKING	Output spool full — Optional odd or even parity while reading Tape breakage — Odd parity check on all characters punched
FEATURES	Removable plugboard expands flexibility by providing ability to control input data format, check odd or even parity, delete specified characters, stop operation or indicate end of file upon detection of specified characters, and perform various logical functions. Extensive error monitoring for high-accuracy data transfers. Photoelectric reading mechanism for accurate, reliable reading without tape wear.

CARD PUNCH AND CONTROL (CPZ201)



DATA MEDIUM	Standard 80-column with round or square corners.
DATA FORMATS	Standard punched card code, edited decimal and 12-row binary.
SPEED	300 cards per minute.
OPERATIONAL MODE	On-line decimal or binary and edited Hollerith.

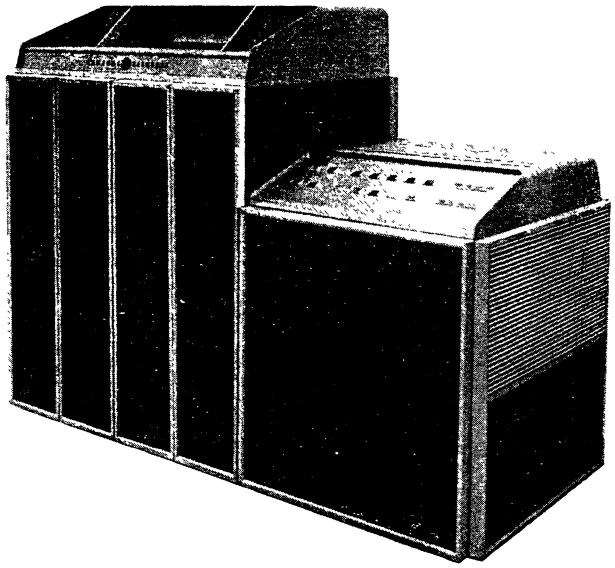
CHECKING

Card feed	Stacker full
Card synchronization	Chad box not properly inserted
Parity	Chad box full
Card jam	Read-after-punch
Hopper empty	Auxiliary stacker full

FEATURES

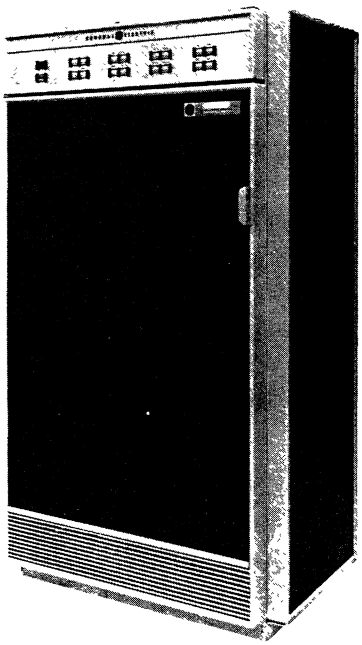
1200-card stacker and 1200-card hopper capacities.
An auxiliary stacker of 100-card capacity. Cards are automatically directed to the auxiliary stacker when punch errors are detected.
Continued operation while loading or removing cards.
Extensive error monitoring for high-accuracy data transfers.
Automatic delay turnoff and halt.

ASCII EXTENDED CHARACTER SET PRINTER (PRT202)

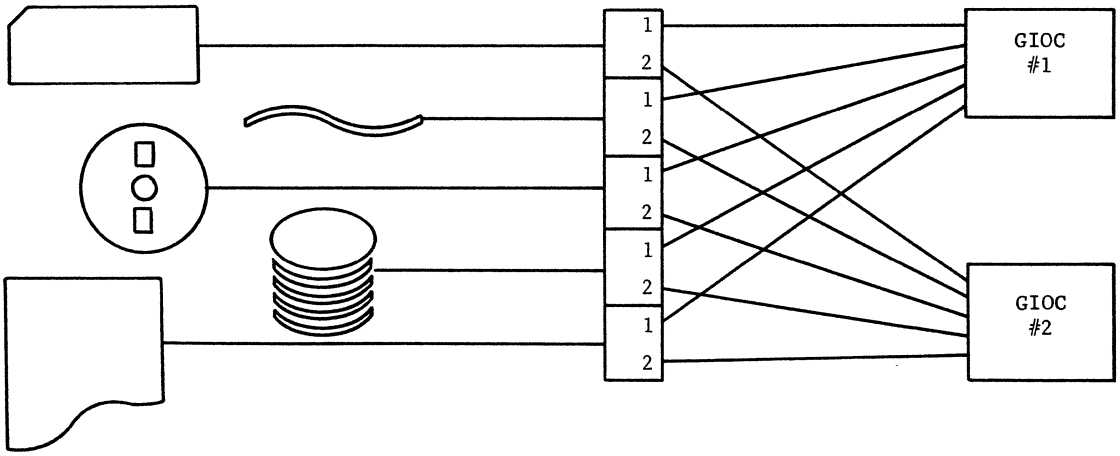


DATA MEDIUM	Continuous forms, 3 to 19 inches wide, up to 22 inches long, 1 to 5 copies (original and 4 carbons).
DATA FORMATS	Standard ASCII font. Vertical spacing is 6 lines per inch.
SPEEDS	600 lpm, printing all 94 ASCII printable characters, 1200 lpm, printing a subset of 34 most used characters.
OPERATIONAL MODES	Edit mode allows column skipping and special slewing by countdown or VFU loop. Nonedit mode suppresses special editing functions.
CHECKING	Parity on input data and on VFU tape punch configurations.
FEATURES	Photoelectric sensing of the VFU tape for reliability. Switches for communication between operator and control program to position magnetically recorded input media. Separate VFU mechanism for each mode of vertical line density. Programmed control of slewing by VFU tape or countdown, includes top-of-page slew.

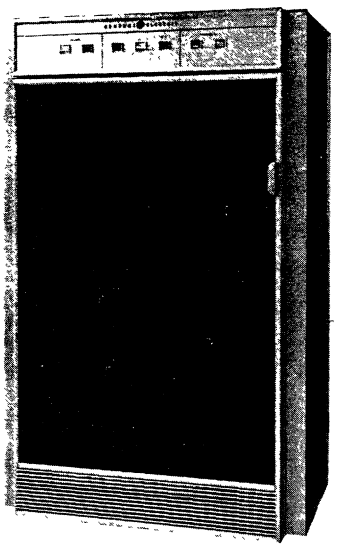
PERIPHERAL SWITCH CONSOLE (PSC200)



The peripheral switch console can switch peripherals and GIOC's in a multiprocessor system. This permits easy and rapid configuration of systems and permits convenient equipment substitution for maintenance purposes. The subsystem consists of a console which contains operator switches and from 1 to 16 switch units. Each unit can be used to connect a single peripheral subsystem to either of two GIOC's, or either of two subsystems to one GIOC. Units are switched by the operator. Buttons on the control and indicator panel of the console control the switch units.

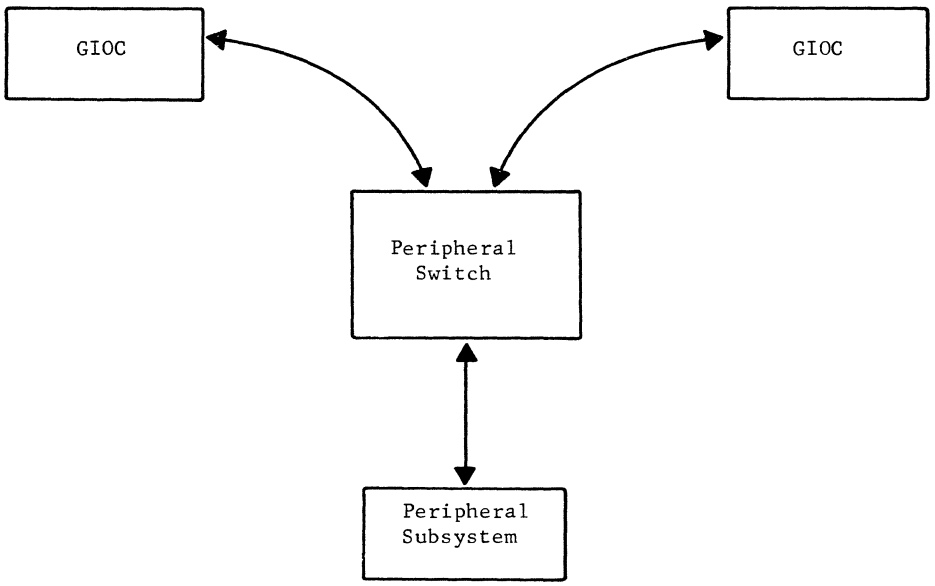


PROGRAMMABLE PERIPHERAL SWITCH (PS6010)



The programmable peripheral switch allows two GIOC's to share the use of the same single-channel peripheral subsystem. It is used most often with storage subsystems such as the Mass Storage Unit, MSU388.

The peripheral switch can switch one peripheral subsystem from one GIOC to another under program control. Switching can also be done manually by the operator.



11. MULTICS-645/I

The multi-user environment of the GE-645 is controlled by the Multics-645/I operating system. Some of the salient features of the operating system have been described briefly in chapters 3 and 4. In this chapter the general functional organization of Multics-645/I and some of its more detailed characteristics are explained.

As the controlling element of an information processing utility, Multics-645/I possesses some characteristics which have not been incorporated in prior operating systems. These are its ability to be maintained while it is in useful operation, and its high degree of flexibility which enables it to be adapted to the needs of a wide variety of users.

These unique characteristics are attainable because of the design philosophy used in the implementation of Multics-645/I. This is the philosophy of a distributed operating system. In this philosophy the operating system does not exist as an entity apart from its users; rather, it becomes part of each of its users' programs.

The way in which the operating system is attached to user programs is controlled to some degree by the users. This enables a user to substitute a module of his own in place of the standard module provided by the operating system. In this way the user can obtain a high degree of functional specialization without sacrificing the general functional capabilities of Multics-645/I.

General users are restricted in varying degrees in the extent to which they may substitute their own modules for modules supplied by Multics-645/I. In this way, damage to the system or to other users is prevented. However, system programmers with the responsibility for maintaining the system software may obtain the level of privilege needed to modify any but the most central and sensitive modules of the system. In this way test versions of system modules can be exercised before being placed in general use and without the need for removing the system from use while they are being tested.

The four major functional subdivisions of Multics-645/I are: (1) the supervisor, (2) the command system, (3) the file system, and (4) the input/output system. The use of these will be illustrated in an example; each will be described in the remaining sections of this chapter.

USER INTERACTION WITH MULTICS-645/I

This example discusses the principal steps that take place as the user at a terminal writes a program, compiles it, and executes it. The example is presented to help the reader understand the overall functioning of the major parts of Multics-645/I.

The user begins by dialing the number of the utility center. The supervisor reacts by logging in the user and building various segments that are needed by the operating system.

Now the user is ready to type in his program. What he actually does at this point is build a file of source language statements. A compiler operates on this file. In building the file, the user types in a command that tells the system he wants to build a file. The command system executes his command by giving control to an editing routine. He now types his program. As he types, the file system is placing his input in a file. The decisions of which pages to use in core memory for his file are made by the file system. Control of the input/output hardware at the utility center is handled by the input/output system under the direction of the file system.

When the user finishes typing his program, he types a command that directs Multics-645/I to compile his program. The command system gives control to the compiler.

The compiler uses the file system to obtain the file of source language statements. The file system, and through it, the input/output system, is used to obtain additional parts of the compiler itself from secondary storage as the compilation proceeds. The principal output from the compilation is the object program. It will be in a file that is built as the compiler delivers its output to the file system. A message is typed when the compilation is done. The user then types a command that directs the system to execute his program. (Assume that this program reads some data from the user's terminal, performs some calculations, and types the results on his terminal.) The command system interprets his execute command and, through the supervisor, causes his program to become a candidate for execution. When his program is placed in execution, it asks him for the input data. At this point, the program does not actually need a processor and the supervisor will assign the processor to some other user. While he is typing his data, the I/O system is taking care of the details of reading the input data.

When he is finished typing the input data, he indicates this to the system and an interrupt occurs. The supervisor receives control, analyzes the interrupt, and gives control to the input/output system. It verifies that the data was received satisfactorily and notifies the supervisor that the user's data is now available. The supervisor now gives control back to the program, and the program performs its calculations. The results are now ready to be sent to the user. His program calls on the input/output system to type the results.

To terminate his run, the user initiates the logout procedure. The supervisor summarizes resource utilization information for accounting purposes and takes the necessary steps to terminate the user's run. As time passes, and his program and data files are not used, the file system removes his files from memory to make room for active users. This example has merely touched on some of the functions performed by the various parts of the operating system. More information on the supervisor, command system, file system, and input/output system is supplied in the following sections of this chapter.

SUPERVISOR

This module of Multics-645/I controls and measures the use of the processors. Control is exercised by interrupts and faults. Hence, Multics-645/I is said to be an interrupt-driven operating system.

A fault or interrupt causes a processor to enter the supervisor. In some cases, as in a serious hardware malfunction, this results in immediate entry to an appropriate recovery routine of the supervisor. Far more frequently, the fault or interrupt signals a normal occurrence.

Examples of such occurrences are:

1. A processor attempts to refer to a page not in core memory.
2. An input/output operation has terminated on a peripheral device or remote terminal.
3. The memory access counter has gone to zero, signifying that a user has used up his current processing allowance.

In each of these cases and many others, the occurrence of a fault or interrupt is a signal to the supervisor that the status of some program has changed.

Such changes in status imply to the supervisor a need to perform some action. In the examples given above the actions might be:

1. Prevent execution of the program whose page is not in core memory. Call the file system to retrieve the needed page.
2. Call the termination routine in the I/O system for the device which has terminated. This routine in turn checks to see if the termination was successful and, if so, attempts to start other queued input/output on the same device.
3. The supervisor starts another program since the current one has exhausted its processing allowance.

The traffic controller maintains overall control of the system. It views processors as belonging to an anonymous pool and allocates them to the tasks to be performed in such a way that all tasks are performed within their required response time. It allocates processor time to real-time demands in time to avoid loss of data or control, allocates resources to interactive users to allow them to proceed without delay, provides enough service to utility programs to keep such operations as tape-to-printer operating at full printer speed, and moves batch programs ahead fast enough to complete them at a predetermined time of day.

The traffic controller consists of three major components: the dispatcher, the sequencer, and the scheduler. The dispatcher performs the detailed initiation of tasks using the available devices of the system. Its data base is a list of tasks to be performed, ordered in priority sequence. Since tasks are normally constrained to a short time duration, the dispatcher is entered frequently. However, the execution time in the dispatcher is short, since all it is requred to do at the termination of one task is to account for the execution time of that task and then initiate the task at the top of the task list.

A program is either running (if it is being executed by some processor), ready (if it could be running but is not because a processor is not assigned to it), or blocked (if it is awaiting an external action such as completion of an I/O operation). Only ready tasks are entered into the dispatcher's task list. A program changes from one of the other states to ready as a result of a previously unsatisfied external condition becoming satisfied (in the blocked case) or when a running process is discontinued as a result of a memory access counter runout. Any of these changes in status are signaled by a fault or interrupt.

When a fault or interrupt occurs, the processor conditions are stored in a special segment. The task which is to respond to the fault or interrupt is then entered into the dispatcher task list in correct priority order by the sequencer. A system routine exists to respond to each fault or interrupt, but users may specify that they wish their own routines to be used instead. Hence, either a system or user task is entered into the task list by the sequencer for each fault or interrupt that occurs.

The remaining component of the traffic controller is the scheduler. It determines and periodically adjusts the priorities assigned to tasks based upon (1) a human's prespecified judgment of their relative importance, (2) their present degree of completion versus what was expected by this time, and (3) effects of priority changes on system utilization and workload. The scheduler optimizes the use of the system on the basis of certain design criteria. Such criteria, however, tend to depend upon local objectives, practices and policies, so that it is expected that many installations will substitute their own schedulers at delivery time or shortly thereafter. Substitutions of this type are anticipated and pose no serious problems so long as Multics-645/I standards are adhered to.

The supervisor provides a means of protecting the modules of Multics-645/I from unauthorized entry by system users. The protection is provided in a hierarchy which allows users with increasing degrees of authority to gain greater and greater access to the system modules. This gives rise to the concept of rings of protection in which all users are allowed access to the outermost ring, a smaller number to the next-inner ring, etc., until the most deeply nested ring is accessible to only a few highly privileged system programmers.

As control of the processors is switched from one program to another, the amount of processor time used by each program is accumulated. This metering of processor time is used as a method of determining the charges for each user and as a source for analyzing the performance of the system.

COMMAND SYSTEM

Commands are calls to programs expressed in user-oriented language. Commands are often issued from the remote terminals of interactive users, but they may also be issued by other files for controlling batch processing applications.

In providing a general interactive capability, it is important that every user not be required to become expert in the detailed programming techniques used in communicating with terminal devices. When using the Multics-645/I command system, one can enter information from a terminal in a simple and standard manner. Programs are not unduly complicated by virtue of being interactive. The command system is written to help the user who is apt to make typing errors, forget arguments, and make other mistakes. The system can often "understand" the user's intentions in spite of the fact that he makes errors or states his intentions in an obscure manner. When the system does not "understand" what is entered from a terminal, it types out statements which lead the user to provide the necessary and correct information.

Two essential parts of the command system are the "listener" and the "SHELL." The listener reads and controls the storage of messages arriving from remote terminals. The SHELL translates these console messages into calls for procedures. As its name implies, the SHELL is an interface between the user and the procedure being called.

The listener reads the message into a buffer. When it detects the end of the message, it calls the SHELL, using the buffer contents as arguments. The SHELL then translates the message into a call to a procedure. The SHELL breaks the character string of the message into substrings according to the syntax for commands. It assumes the first substring to be a procedure name and interprets the remaining substrings as arguments (data) for that procedure. The SHELL converts the argument string to the form expected by the procedure named. When an invalid argument is detected, the user is informed of the error (by a typed message) and asked to take appropriate corrective action, for example, to retype the argument.

After the SHELL has the argument in the form expected, it creates a standard call for the procedure named and then transfers control to it. This procedure now has control until its execution is completed. Control is first returned to the SHELL, and then to the listener to receive the next message.

The SHELL can call any procedure for the user, providing the syntax of the command language has not been violated, and the procedure called is in a file which the user has permission to execute.

The SHELL is itself a system subroutine. It may be called by any procedure. Therefore, it may be called recursively through any depth of nesting. This allows it to be used by any procedure to perform the standard terminal interface functions.

When a user types a command, he may want to include parameters which control the way a called program is executed. For example, a call to a compiler might contain a parameter to specify whether a listing file is to be created. Parameters of this type are called meta-arguments. Meta-arguments are most frequently used to change the setting of a system option. System options may be set permanently for a particular user or they may be set only for the duration of an interactive session.

Although the SHELL may call any procedure specified by the user, it is used mostly for calling system commands. A system command is a procedure available to all users. It is maintained by system programmers and appears in a special directory. This allows the user to call the utility procedures of the Multics-645/I system with minimum effort.

FILE SYSTEM

The file system of Multics-645/I enables the user to symbolically refer to any segment, file, or the contents of any segment or file. Furthermore, it enables him to do this in a device-independent way. When interacting with stored information, he receives the impression he is operating in an environment consisting of a great many independent high-speed memories, each of which can be referred to by name.

A file is an ordered sequence of elements in which an element is a machine word, character, or bit. In Multics-645/I, all segments are files, and all files are either bounded or unbounded. A bounded file has up to 2^{18} words, and can therefore be read into core memory as a single segment. An unbounded file has no limit to the number of words it may contain; it is viewed through a "window" which is 2^{18} words wide. At any one time, this window can be placed over any part of the file. Therefore, all parts of the unbounded file can be referenced by moving the window over the length of the file.

A customary file system provides the means for:

1. Accepting a named file from a user.
2. Assigning the file to machine addressable storage.
3. Retrieving the file by its symbolic name.

Functions of Multics-645/I File System

In addition to these usual functions of a file system, (i.e., accepting, storing, and retrieving named files) the file system of Multics-645/I provides for:

1. Detecting a frequently-used file and moving it to a storage medium with a faster access time.

2. Detecting a seldom-used file and moving it to a storage medium with a slower access time.

3. Guarding all private files from unauthorized viewers.

4. Allowing a user to share his files with one or more other users.

5. Establishing common files.

6. Protecting all files from misuse or accident.

Controls

The file system controls the dynamic assignment of information to on-line storage (both primary and secondary). A finite amount of information is kept on drums, discs, magnetic cards, or tapes (secondary storage). A user does not need to know how or where his file is stored. No area of storage is permanently assigned to any one user.

The file system stores all segments as files. Some of these segments are directories. A directory is a special file containing a list of entries, in which each entry is a pointer to another file. An entry which points directly to a file is called a "branch", while an entry which points to a file via another directory entry is called a "link". A chain of links eventually terminates in a branch. All directories are maintained by the file system itself.

A branch contains a description of the way in which the file may be used and the way in which it is being used. The description includes information such as:

1. The physical address of the file.

2. The time the file was created or last modified.

3. The time the file was last referred to.

4. Access control information for the branch.

5. The current status of the file (open for reading by N users, open for reading and writing by one user, open for data sharing by N users, or not available).

Associated with a link is the pointer to another directory. This pointer is a symbolic name that uniquely identifies the linked entry within the hierarchy. A link derives its access control information from the branch to which it effectively points.

Storage

A secondary storage device is assigned a level number by Multics-645/I based on the device's relative speed. The device with the highest transmission and access rate is assigned the highest level number. When a file is to be used, it is automatically moved to the highest-level storage device available. This process is tempered by considerations such as the size of the file and the frequency of use.

As more space is needed on a particular storage device, the least used files are moved to a lower-level storage device. Files not being used continue to be moved to lower-level storage until the desired amount of higher-level storage is available. If a file must be moved to a lower level from the lowest-level on-line storage device, the file is placed on tape and the tape reel is stored. This fact is noted in the appropriate branch of the file's directory.

The file system can work with many configurations of secondary storage devices which may cover a wide range of speeds and capacities. Thus, user programs and system programs are independent of the configuration of secondary storage.

When a user first logs in at a terminal, he gives the file system his account name or number. Subsequently, all files created by this user are labeled with his account name. The accounting procedure maintains records of all storage use and storage allotments. A storage allotment is the amount of information that a particular account is allowed to store on-line at any one time.

When an account approaches or exceeds its storage allotment, the accounting procedure notifies the file system that the account is almost empty or is overdrawn. The action that is taken when an account is overdrawn can be tailored to the needs of a specific user or group of users. When a user wants to increase his allowed use of secondary storage by a specific amount, the file system gives the accounting procedure his account name with the amount and class of storage requested.

Basic File System

The basic file system is the part of the software that manages segments and pages. A segment is made available to a program when the program refers to a file through the use of read and write statements or by means of segment addressing. The basic file system performs the following functions:

1. Maintains directories of existing segments.
2. Makes segments available upon request of a person or program.
3. Creates new segments.
4. Deletes existing segments.

A block diagram of the modules that make up the basic file system is shown in Figure 33. Some of the data bases that are common to the modules are shown as circles in the diagram. These are:

CORE MAP	Core Map
DIRECTORIES	Directory Data Base
SNT	Segment Name Table
AST	Active Segment Table
AFT	Active File Table

In this diagram, the directional lines indicate the flow of control through the use of formal calling sequences, with formal return implied. Lines with double arrowheads indicate possible flow of control in either direction. The modules and data bases shown below the dotted line reside in core memory at all times. The modules of the file system are individually described on pages which follow. The modules are:

Segment control	Access control
Page control	Usage control
Directory control	Core control

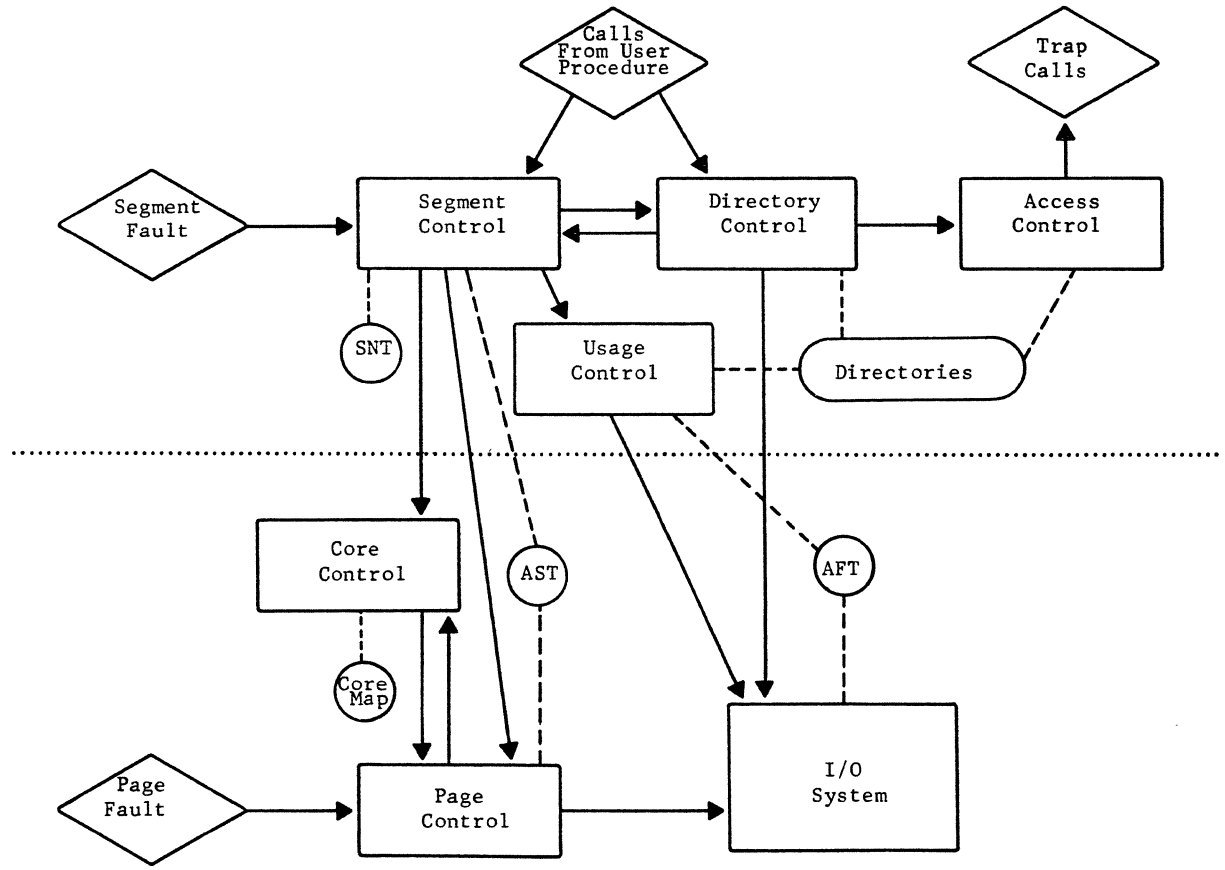


Figure 33. The Basic File System

SEGMENT CONTROL. Segment control maintains records in its segment name table (SNT) of all segments referenced by a user program. There is a different SNT for each user. When a user refers to a segment for the first time, directory control is called. It finds the branch of the appropriate file in secondary storage and in turn calls segment control.

Segment control then takes the following steps:

1. Assigns a segment number to this segment.
2. Creates a new entry in the segment name table indicating that this segment is now known to the given program.
3. Opens (or reactivates if already open) the file or files which are to receive I/O requests resulting from paging.
4. Creates or updates the appropriate entry in the active segment table.
5. Determines the descriptor control bits for the segment from the effective mode of the file branch.
6. Establishes a page table and the remainder of the segment descriptor for this segment if the segment is not already active for some other process.
7. Returns the segment number to the calling procedure.

When a page table is initially constructed by segment control, each entry in that page table contains a fault indication and a flag indicating what action should be taken if and when that fault occurs. This flag may indicate one of the following actions:

1. Assign a blank page.
2. Retrieve the missing page from the original file.
3. Retrieve the missing page from a working copy of the original file.

In addition to the functions described above, segment control provides entries through which the user may ask questions or make declarations involving the use of segments known to his process. Some of these functions are:

1. Declare that some specific locations within a segment are no longer needed at this time.
2. Declare that some specific locations within a segment are to be reassigned. (The user is about to overwrite these locations.)
3. Declare that some specific locations within a segment are to be required shortly.
4. Terminate a segment and indicate that this segment is no longer to be considered as known to this process.
5. Truncate a segment.

DIRECTORY CONTROL. Directory control provides all the basic tools for manipulating entries within a directory. It may be called by segment control or directly as a result of a file system command. This module performs primitive operations that are usually augmented by more elaborate system library procedures. The following is a list of some of these procedures:

1. Create a new directory entry.
2. Delete an existing entry.
3. Rename an entry.
4. Return status information concerning a particular entry or entries.
5. Change the access control information for a particular branch.
6. Find a branch and assign a segment number to it for the requesting user.

Whenever a user wishes to perform any operation on the contents of a file pointed to by a particular directory entry, the branch pointing directly to that file is first obtained from that directory. Access control is then called to determine what the user is permitted to do with the file.

ACCESS CONTROL. Access control is called by directory control to evaluate the access control information for a particular branch. Thus, the name of the user and the access control information for the branch in question is made available to access control. It is also given a code indicating the type of operation (e.g., rename, delete) which is being attempted. The access control returns a single effective mode to directory control. The effective mode is the mode which governs the use of a file with respect to the current user or program. This effective mode is used by directory control to determine if the requested operation is to be permitted.

USAGE CONTROL. Usage control opens and closes files for segment control. A file may, for example, be open for reading by n users, writing (i.e., reading and writing) by one user, or for data sharing by n users (i.e., it is used as a common data base). The specified use of a file determines it usage status. If pages of the file (segment) are going to be brought to core memory, the file must have an entry in the active file table (AFT), in which case the file is said to be active. This entry provides sufficient information to control subsequent I/O requests for that file.

If a file is closed, opening the file for the current (would-be) user entails setting the file usage status in the directory and making the appropriate entry in the AFT. If the file is already open (on behalf of other users), opening the file for the would-be-user entails comparing the requested usage status with the existing usage status. If the requested status is incompatible with the present status of the file, the current process must be blocked. For example, if the current user wishes to read a stable copy of the file and there is at present a user writing into that file, the requested status (reading) and the present status (reading and writing) are said to be incompatible.

If the requested status and the present status of the file are found to be compatible, the file is opened for the given user. If no AFT entry currently exists at this time, a new AFT entry is created. The directory branch is updated and the pointer to the corresponding entry in the AFT is returned to the calling procedure. This pointer is used to direct requests for subsequent input or output to the correct file.

Whenever a file is opened, usuage control may decide to move the file to a higher or lower level on-line storage device, as determined by the multilevel storage algorithm.

PAGE CONTROL. Control passes to page control by means of a missing-page fault in a page table in use by the current program. This fault may indicate that a new page should be assigned or that an existing page should be read in from an active file. In either case, a block of core memory must be assigned before anything else can happen. This is accomplished by a call to core control.

If a new page is to be read in, the page table entry for the missing page contains a pointer to the appropriate entry in the active segment table. This pointer is passed to the I/O system with a read request to bring the correct page to core memory. Page control may also be called by segment control to perform input or output operations for active segments, or by core memory control to remove pages of active segments.

CORE CONTROL. Core control maintains information on the availability of each 64-word block of core memory and how this block is being used. This information is kept in the core map, and includes the following: (1) the availability of the block to the system, (2) whether the contents of the block may be removed from core memory (and its eligibility for removal), and (3) whether the contents of the block may be moved elsewhere in core memory. The core control module provides core memory space for pages to the page control module and provides core space for page tables and contiguously paged segments to the segment control module.

Blocks whose contents are candidates for removal are indicated in the core map by a usage list in the order of decreasing frequency of usage. This is a linked list in the core map. At periodic intervals, the list is relinked by core control on the basis of use in that period. When blocks of core memory are required for assignment, they are selected from a pool of unassigned core blocks. The pool is maintained so that requests may normally be satisfied simply and quickly. This supply of unassigned core is replenished by periodic removal of infrequently used pages.

Linker

The Linker is the part of the file system which converts symbolic references between segments to hardware-intelligible references at execution time. In previous systems, communication between subprograms was set up for an entire program prior to execution. This was done for all subprograms whether or not they were ever executed. When operating under Multics-645/I, communication between segments (linking) is established only if and when a reference occurs between those segments.

LINKAGE SECTION. When a segment is created, for example, by a compiler, specific information is saved in addition to the text of the segment. This information is called a linkage section. The segment with which it is associated is called its owner segment.

A linkage section contains the following:

1. A pair of words for each reference to a symbol outside its owner segment. This pair is called a link-pair.
2. The symbol associated with each link-pair. This is, in general, a two-component symbol specifying a segment name and a symbol within the named segment.
3. The name and relative location of each symbol within the owner segment which can be referred to from outside the owner segment.

LINKAGE SEGMENT. In addition to various other segments, each program has a linkage segment. It consists of the linkage sections of each segment of the program. As new segments are referenced during execution, additional linkage sections are added to the linkage segment.

A pair of bases designated lb and lp are associated with the linkage segment. The lb contains the segment number of the linkage segment and the lp is an internal base which is set to point to the linkage section of the segment in execution.

LINKING. All references to symbols outside the owner segment are made indirectly via a link-pair in the linkage section. Each link-pair is given an initial value which causes the processor to fault when it is referenced. This fault causes the linker to be activated. When entered, the linker does the following:

1. Obtains the name of the referenced segment from the linkage segment.
2. Enters segment control to obtain the segment number of this segment for the requesting user.
3. If the segment name is not known to segment control, it is obtained by entering directory control.
4. Adds the linkage section of the referenced segment to the linkage segment (unless it is already there).
5. Creates an SDW for the referenced segment (unless it is already there).
6. From the newly added linkage section, determines the relative location of the symbol in the referenced segment.
7. Enters the segment number and relative location of the reference in the link-pair and places an ITS modifier in its first word. (ITB is used in some cases.) Hence, on subsequent references, the linker will not be entered.
8. Returns control to the faulted program.

I/O SYSTEM

The input/output (I/O) system of Multics-645/I is used by the file system and also directly by the user. The file system calls upon the I/O system when files need to be moved within the secondary storage hierarchy. When the user's program needs to transfer information between the program and a peripheral or terminal, the program calls upon the I/O system to perform the actual data transfer. As in the file system, the user is allowed to operate upon a file without concern for the device upon which it is being written or read.

A particular file may be assigned by the user to one device on one day and a different device on another. Further, an output file can be assigned to several devices so as to be broadcast to various locations. This device interchangeability is achieved without modification to the user's program.

The user communicates with the I/O system using high level language calls. He works with logical records. The I/O system groups these into physical records in a manner which is most efficient for the device to which the file is assigned.

In the event of errors or exception conditions in the device, the I/O system performs appropriate action. The user may supply his own error recovery and exception processing routines if the standard versions provided by the system do not suit his needs.

In performing input/output on files, which may be assigned to a variety of devices, the input/output system performs code conversion, queuing, and buffering. It performs these in a manner determined by the device involved.

When an I/O command is received by the I/O system, it performs the necessary code conversion and buffering as a function of the device to which the file is assigned. When it is time to read or write a physical record to the device, the I/O system creates a data control word list to control the reading and writing of the information. In constructing the data control word list, the I/O system takes account of the fact that the information to be transferred may be dispersed throughout the paged memory environment of the GE-645.

When it is time to issue a command to a device, the I/O system determines if issuing the command would result in overloading the GIOC or the device subsystem. If for this or any other reason the command cannot be issued immediately, it is queued. In any event, the I/O system ultimately issues the command to the device subsystem.

The I/O system allocates input/output resources. It assigns entire subsystems such as card readers and printers to individual programs as well as allocates parts of systems to individual users. For example, the I/O system maintains records of the amount of storage available for assignment and the amount already assigned on each mass-storage device.

STANDARDS

All modules of Multics-645/I and all user programs communicate with each other in the same way. Powerful communication standards are universally employed to make this possible. Some of the most significant communication standards are described in the following paragraphs.

Each active program operating under Multics-645/I possesses a minimum set of segments. Some of these are: a descriptor segment, a linkage segment, and a stack segment.

The descriptor segment defines the segments accessible to the program. (Its functions are described in chapter 6.) Associated with each descriptor segment is a descriptor segment page table and a descriptor base register setting.

The linkage segment contains information which allows symbolic references between segments to be converted to hardware-intelligible addresses at execution time. It also contains definitions of the modes of representation of the arguments used by the segment. This information allows

automatic conversion of arguments when the segment is called. The segment number of the linkage segment is stored in an address base register (ABR) known symbolically as lb. The linkage information for the segment currently in execution is pointed to by a paired ABR labeled lp.

Every program is assigned a stack segment which may be used for temporary storage. Its use in conjunction with the use of pure procedures is very convenient since pure procedures have no changeable storage to use for individual purposes.

A pair of bases known symbolically as sb and sp are assigned for the purpose of maintaining the stack segment as a push-down stack. The base sb locates the base of the stack while sp provides an origin for its current working level. A pointer to the base of the previous working level is stored at a known location below the current base so that the stack can always be returned to any lower level of storage. Figure 34 contains an example of a stack segment.

	...	
	...	temporary storage for alpha
	...	
sp when in beta	0-7	save bases when calling gamma
	8-15	save registers when calling gamma
	16-17	value of sp when in alpha (last sp)
	18-19	value of sp when in gamma (next sp)
	20-21	return when calling gamma
	22-23	(used by execute-only/master procedures)
	...	
	...	temporary storage for beta
	...	
sp when in gamma	0-7	save bases when calling delta
	8-15	save registers when calling delta
	16-17	value of sp when in beta
	18-19	value of sp when in delta
	20-21	return when calling delta
	22-23	(used by execute-only/master procedures)
	...	
	...	temporary storage for gamma
	...	
sp when in delta	0-7	...

Figure 34. Stack Segment as Alpha Calls Beta, Beta Calls Gamma, and Gamma Calls Delta

All procedures operating under Multics-645/I are called as closed subroutines. Standard call, save, and return macros which use the stack segment facilitate this mode of communication. Each calling procedure uses as much temporary storage as it needs relative to the current value of sp.

If one procedure calls another, the standard call macro is used. The call macro saves the processor conditions in the stack at the base of the area that the current procedure has used for temporary storage. (See Figure 34.) In addition to saving the machine conditions at the point of executing the call macro, the call sets a base pair to an argument list. The base pair is known symbolically as ab and ap, where ab points to the origin of the list and ap points at the argument currently being referenced. After performing these functions, the call macro transfers to the called procedure.

The save macro is located at the entry point of the called procedure. It moves the stack pointer, sp, to a point above the area used by the calling procedure. It also stores pointers to the base of the old stack area and the beginning of the next one to be used. This is in case the current procedure again performs a call.

The called procedure ultimately performs a return unless the program which contains it is terminated. When a return macro is executed, the processor registers are returned to their condition at the time that the corresponding call macro was executed. However, the instruction counter will have been incremented to point to the instruction immediately following the last word of the expanded call macro.

Notice that the calling procedure need make no assumptions regarding the use of machine registers. They are returned to their original contents after the call and return have been executed. Also, a call to a pure procedure which uses the argument list and stack segment for its temporary storage can be made recursively to any depth, either directly or through a chain of other calls.

12. MULTICS-645/I RELATED SOFTWARE

The availability of system software is of vital interest to a user. The GE-645 standard software is being developed in a way that permits every user to easily progress from the environment of the GE-625/635 to the environment of the GE-645 (Multics-645/I).

The GE-625/635 computer systems improved total system utilization through the use of multiprogramming and multiprocessing. The GE-645 retains these advantages and adds to them shared direct access and storage management capabilities which solve the vital problem of user access to the system.

GE-645 COMPATIBILITY

Compatibility of the 600 line of General Electric systems goes beyond the mere hardware instruction compatibility. To avoid any conversion difficulty between the operating systems, the complete GE-625/635 operating system is available on the GE-645. The system is GECOS-II which is a multiprogramming/multiprocessing operating supervisor. Programs written for the 625/635 system are assured of running on the GE-645.

INTERACTIVE MODE

The direct access mode of operation is often referred to as the interactive mode. In the interactive mode, the user can communicate directly with the system from a remote terminal.

This is a simplified inquiry-response mode which requires little or no technical assistance. The specialist or user need not be technically oriented to use it to solve data processing and scientific problems. He requires only minimal instruction. The advantage in simplicity and speed of operation in using the conversational mode is shown in Figures 35 and 36.

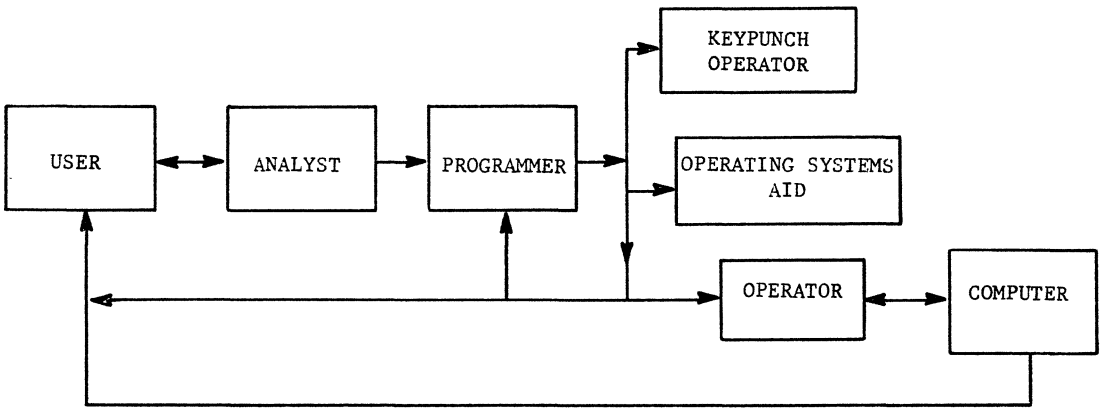


Figure 35. Normal Flow of Operations Before Direct Access

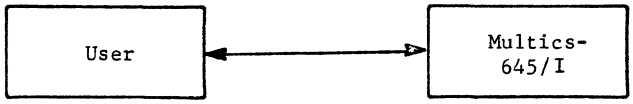


Figure 36. Direct Access Flow of Operations

LANGUAGE PROCESSORS

Language processors used under Multics-645/I are interactive. Initial language processors are:

1. FORTRAN IV - Source language compatible with FORTRAN IV on the GE-625/635.
2. ALGOL - Source language compatible with ALGOL on the GE-625/635. Some extensions are made to give ALGOL character string capabilities.
3. PL/1 - Several PL/1 processors are to be developed.
4. GE-645 Assembler - Has extensive macro capabilities that allow the users to use the machine language of the GE-645.
5. BASIC* - Beginners All purpose Symbolic Instruction Code. It resembles ordinary mathematical notation with simple vocabulary and grammar.
6. COBOL - Common Business Oriented Language for business data processing.
7. JOVIAL - Language based on ALGOL and particularly applicable to military command and control applications.
8. GIFT - Translator for translating FORTRAN II to FORTRAN IV.

*Developed by Dartmouth College.

After compiling a segment that was typed in at a terminal, many error messages are sent to that terminal. The maximum permissible size of these interactive source programs can be determined at each installation. Programs can be compiled by batch processing, with error messages and object code sent to secondary storage. The object code generated by these processors can reference remote terminals at run time. The maximum size of the object code programs is essentially unlimited. The object code can also be executed referencing properly constructed files rather than remote terminals. The code reads and writes data on any stored medium at run time.

APPLICATION PROGRAMS

Initial application programs are:

- Linear Programming
- Automatically Programmed Tools (APT)
- APT Postprocessors:
  - Monarch and Jones and Lamson
  - Burgmaster
  - Milwaukee-Matic (Models II and III)
  - Bullard
  - Giddings and Lewis
- Math Routines
- SIMSCRIPT
- Media Conversion
- BMD Statistical Programs
- PERT/TIME
- PERT/COST
- Integrated Data Store (IDS)
- Sort/Merge

SIMULATORS

- IBM 1401
- GE-225

TEST AND DIAGNOSTICS

Hardware test and diagnostic programs are included as an integral part of the software. These test and diagnostics are automatically scheduled and executed at regular (or at planned irregular) intervals or when a need for them becomes apparent as a result of malfunction.

APPENDICES

A. ASCII CHARACTER SET

Multics-645/I uses the revised ASCII character set. It consists of 128 7-bit characters which are right justified in the GE-645 with two leading zeros. The two leading bit positions are reserved for future character set expansion. Any devices which cannot create or accept the full character set will use established conventions for representing the full set. There are no meaningful subsets of the revised ASCII character set.

REVISED ASCII CHARACTER SET

Included in the ASCII character set are 94 printing graphics, a space, and 33 control characters. Within Multics-645/I, all the graphics, the space, and 13 of the control characters are given precise interpretations. The remaining 20 control characters are not currently used by Multics-645/I, but may be used at a later date. The graphics in the set are the following:

Upper Case Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Lower Case Alphabet: abcdefghijklmnopqrstuvwxyz

Numerals: 1234567890

Special Characters

! Exclamation Point	; Semicolon
" Double Quote Mark	< Less Than
# Number Sign	= Equals
$ Dollar Sign	> Greater Than
% Percent	? Question Mark
& Ampersand	` Grave Accent
' Acute Accent	[Opening Bracket
(Opening Parenthesis	~ Tilde
) Closing Parenthesis	] Closing Bracket
* Asterisk	^ Circumflex
+ Plus	_ Underline
, Comma	@ Commercial At
- Hyphen (minus)	{Opening Brace
. Period	¬ Overline
/ Slant	} Closing Brace
: Colon	| Vertical Line

CONTROL CHARACTERS

The following conventions define the only standard meaning for the ASCII control characters within Multics-645/I. These conventions are observed by all standard device interface modules (DIM) and by all system software inside the input/output system interface. Since some hardware devices have different interpretations for some characters, the appropriate DIM must perform any necessary translations. Characters noted as "not used" are reserved; future definitions may be made at any time. Until defined, "not used" control characters encountered on output will be printed out with the octal escape convention. Users wishing a different convention for interpretation of a "not used" character must use a non-standard DIM.

MULTICS-645/I Standard Control Characters

BEL — Sound an audible alarm.

BS — Move carriage back one column. The backspace character implies overstriking, not erasure.

HT — Horizontal Tabulate. Move carriage to next horizontal tab stop. On variable-tab machines, in the absence of an explicit agreement between the typist and his DIM, horizontal tab stops are assumed to be at columns 11, 21, 31, 41, etc.

NL — New Line. Move carriage to left edge of next line. Character code 012 (ASCII LF) is used for this function.

CR — Carriage Return. Move carriage to left edge of the present line.

RRS — Red Ribbon Shift.
Character code 016 (ASCII SO) is used for this function.

BRS — Black Ribbon Shift.
Character code 017 (ASCII SI) is used for this function.

VT — Vertical tabulate, move carriage to next vertical tab stop. On variable-tab machines, in the absence of an agreement between the typist and his DIM, vertical tab stops are assumed to be at line 11, 21, 31, etc.

FF — Form Feed. Move carriage to top of next page.

HLF — Half-Line Forward Feed.
Character code 022 (ASCII DC2) is used for this function.

HLR — Half-Line Reverse Feed.
Character code 024 (ASCII DC4) is used for this function.

RHT — Relative Horizontal Tabulate. The following character is interpreted as a binary integer giving a number of horizontal spaces that the carriage moves.
Character code 021 (ASCII DC1) is used for this function.

RVT — Relative Vertical Tabulate. The following character is interpreted as a binary integer giving a number of vertical lines that the carriage moves.
Character code 023 (ASCII DC3) is used for this function.

Control Characters Not Presently Used By Multics-645/I

NUL(000)	SOH(001)	STX(002)	EXT(003)	EOT(004)	ENQ(005)
ACK(006)	DLE(020)	NAK(025)	SYN(026)	ETB(027)	CAN(030)
EM(031)	SS(032)	ESC(033)	FS(034)	GS(035)	RS(036)
US(037)	DEL(177)				

In the following table, positions occupied by unused control character codes are blank.

ASCII CHARACTER SET ON MULTICS-645/I

	0	1	2	3	4	5	6	7
000								BEL
010	BS	HT	NL	VT	FF	CR	RRS	BRS
020		RHT	HLF	RVT	HLR			
030								
040	space	!	"	#	$	%	&	'
050	(	)	*	+	,	¬	.	/
060	0	1	2	3	4	5	6	7
070	8	9	:	;	<	=	>	?
100	`	A	B	C	D	E	F	G
110	H	I	J	K	L	M	N	O
120	P	Q	R	S	T	U	V	W
130	X	Y	Z	[	~	]	^	_
140	@	a	b	c	d	e	f	g
150	h	i	j	k	l	m	n	o
160	p	q	r	s	t	u	v	w
170	x	y	z	{	-	}	|	

B. SUMMARY OF ACRONYMS AND MNEMONICS

ABR	Address Base Register
AFT	Active File Table
APT	Automatic Programmed Tools
AST	Active Segment Table
ASW	Abnormal Status Word
BASIC	Beginners All purpose Symbolic Instruction Code
CCW	Channel Command Word
COBOL	COmmon Business Oriented Language
CPW	Command Pointer Word
CSW	Current Status Word
DBR	Descriptor Base Register
DCW	Data Control Word
GIOC	Generalized Input/Output Controller
IC	Instruction Counter
IDS	Integrated Data Store
I/O	Input/Output
IP	Indirect to Pair modifiers
IR	Indirect then Register
IT	Indirect then Tally
ITB	Indirect To Base modification
ITS	Indirect To Segment modification
LPW	List Pointer Word
MBZ	Must Be Zero
MULTICS	MULTiplex Information and Computing Service
PBR	Procedure Base Register
PDW	Page Descriptor Word
R	Register modification
RI	Register then Indirect
SCW	Status Control Word
SDW	Segment Descriptor Word
SNT	Segment Name Table

INDEX

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO IN U.S.A.