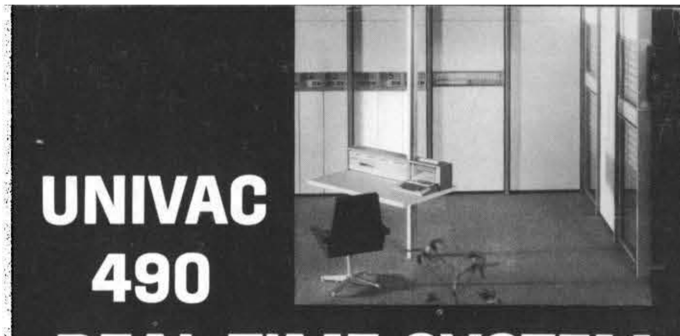
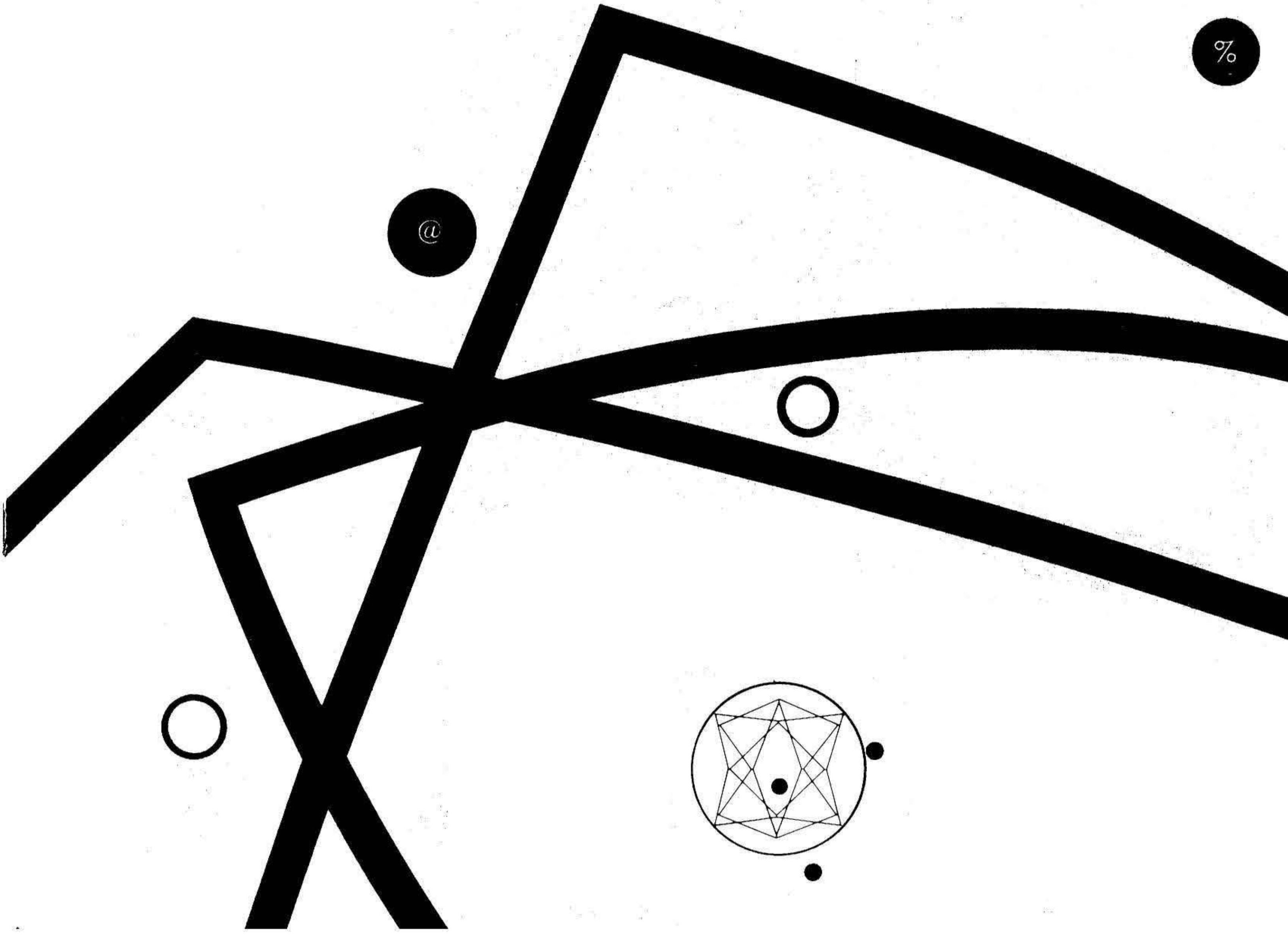


UNIVAC 490 REAL-TIME SYSTEM

GENERAL DESCRIPTION

the UNIVAC® 490 Real-Time System



GENERAL DESCRIPTION

UNIVAC 490 Real-Time System

© 1961 • SPERRY RAND CORPORATION

Contents

1. UNIVAC 490 REAL-TIME SYSTEM

The Real-Time Concept 1
General Characteristics of the Real-Time System 2
High-Speed Communications Linkage 2
Data Storage Facilities 2
Features and Applications 2
  Processing Interrupt 2
  Solid-State Design 3
  Computer-to-Computer Configurations 3
  High-Speed Random Access Storage 3
  A "Time Conscious" System 3
  Incremental Clock 3
  Incremental Interrupt Clock 3
  Day Clock 3
  High Internal Computing Speeds 4
  Equipment Enclosure 4
  Flexible Input-Output Facilities 4
  Automatic Programming 4
  Floating-Point Arithmetic 4
  Programming Checks 4
Special Programming Features 5
  Powerful Instruction Repertoire 5
  Absolute Efficiency 5
  Library of Programmed Routines 5
  Core Storage Search 5
  Wired Memory 5
  Application Versatility 6

2. REAL-TIME COMPUTER

Storage Section 7
  Octal Notation 7
Control Section 8
Arithmetic Section 8
  Arithmetic Registers (Operational Registers) 9
  Transient Registers 10
Operator Console 11
  Computer Control Panel 11
  Console Keyboard and Printer 11
  Wired Memory 11

3. SYSTEM COMPONENTS AND CONFIGURATIONS

Central Site Equipment 12
- Peripheral Units 12
- Peripheral Systems 12
- Magnetic Drum Storage 14
- Magnetic Tape Storage 15
- High-Speed Card Reader 16
- Punch-Verifier Unit 16
- High-Speed Printer 17

Transmission and Communications 18
- Input-Output Channels 18
- Data Transfers 18
- Buffer Mode 18
- Input-Output Control 18
- External Equipment Requirements 19

Remote Input-Output Devices 19
- Keyboard Printer 19
- Uniset Console 20
- The Uniset 21
- Format Control Panel 21

Communications Equipment 21

Communications Systems 23

Party Line Network 23
- Scanner - Selector 23

Party Line Communications System 23
- Transfer Function 24

Line Switching Network (Direct Distance Dialing) 24

Communications Control Unit (Telegraphic Half-Duplex) 24

Communications Equipment for Special Devices 26

4. INSTRUCTIONS

Instruction Word 27

Instruction Cycle 27

Instruction Repertoire 28

Shift Instructions 28

Simple Read Instructions 29

Store Instructions 29

Arithmetic Read Instructions 30

Comparison Instructions 30

Selective Instructions 31

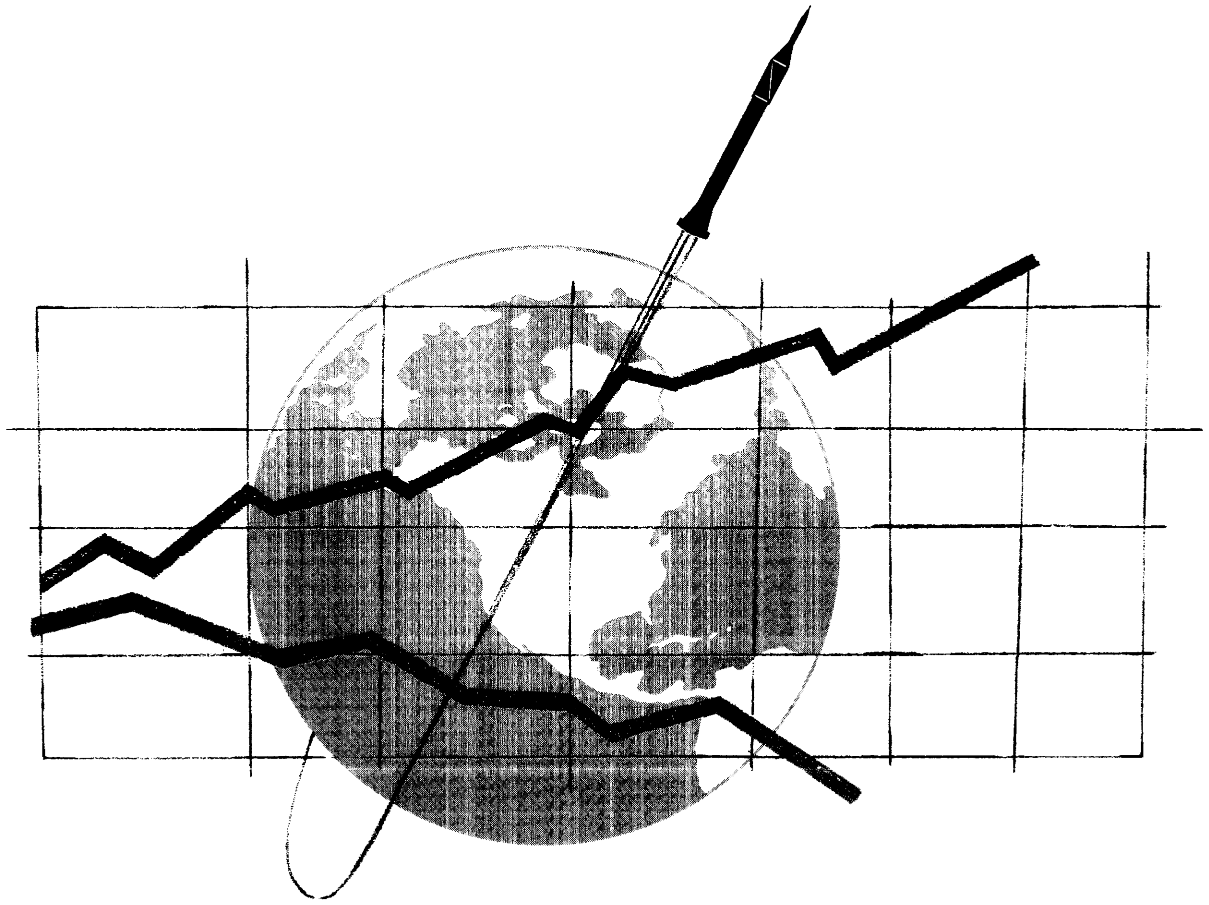
Replace Instructions 31

Jump Instructions 32

Special Program - Modifying Instructions 33

just as feedback is used by a computer

to control a missile's path . . .



. . . up-to-the-minute data from

a UNIVAC Real-Time System can help

influence the course of business curves.

1. UNIVAC 490 Real-Time System

The UNIVAC® 490 Real-Time System is a large-scale, general purpose digital computing system specifically designed for the dynamic organization which has diverse operations demanding stricter control based on real-time computing. This new system extends valuable techniques of real-time processing, long restricted to a limited number of special military applications such as missile guidance, to the broader field of commercial problems.

Remington Rand's installation of UNIVAC Airlines Reservation Systems at several major airlines demonstrated the power of real-time processing. A centrally located UNIVAC File-Computer provides flight information in a fraction of a second to ticket offices scattered throughout the country. The well-documented success of the Airline Reservations System proved the potential of real-time processing and served as the incentive for its full development.

Now, with the Real-Time System, the full commercial potential of real-time processing can become an integral part of your business, bringing with it an efficiency and control only approximated by other devices. In fact, just as feedback is used by a computer to control a missile's path and counteract disrupting forces, up-to-the-minute data from a real-time system can help an organization influence the course of a number of its business curves as they are formed.

THE REAL-TIME CONCEPT

The real-time concept is the fulfillment of management's desire for a method of reversing the direction of a business curve before it gains momentum and attains black-and-white finality. Up-to-the-minute indications of business activity enable the real-time user to detect the suggestion of a downturn and correct it immediately, in much the same way as a guided missile's course is adjusted as it hurtles to its target. If, for example, sales have dipped slightly, the Real-Time Computer will allow management to discover this fact immediately, instead of waiting for quarterly reports. As a result, management's remedial actions are effectively timed. And it is the timely decision which has the greatest significance in the intensified competition of business and industry today. To assure timeliness in organizational activities, it is necessary to utilize a real-time processing system, with its ability to communicate with many remote locations and its large storage facilities, to reflect sales, profits, costs, production, and other pertinent data.

Thus, the real-time concept is a significant advance in data-processing, a field which heretofore employed batch processing only. That is, master data, or information which is altered infrequently or in a known manner, was updated at intervals, upon the accumulation of enough transaction data, or information characterized by essentially random and unpredictable incidence. Real-time processing, however, eliminates the lag between the occurrence of transactions and their postings to a master file. By updating the master file immediately upon receipt of a new transaction from a remote source, the Real-Time System can present a truly current report of the status of any application—a feat impossible for the batch-processing computer with its externally stored master data.

GENERAL CHARACTERISTICS OF THE REAL-TIME SYSTEM

The specific design and function of the remote input-output units of the UNIVAC 490 Real-Time System are dictated by the nature of the individual application. In general, however, input units are able to accept transaction data with speed and reliability while the output units display results with accuracy and clarity. The computing unit of the system is general purpose, and therefore, ideal for all types of applications.

High-Speed Communications Linkage

Common carriers, such as American Telephone and Telegraph, Western Union, American Cable and Radio, afford high-speed communication facilities for two-way transmission of data between the central site Computer and the remote input-output units. Transactions originating at remote points are conveyed along these wires directly to the Computer where they are immediately evaluated and processed. Then the result is returned to the originator and other appropriate distant points, the whole transaction being accomplished in seconds.

Data Storage Facilities

Since the UNIVAC 490 Real-Time System applies transaction data to the master file information as the transaction data occurs, the system employs extensive data-storage facilities which are capable of storing entire master files of information. In addition, these facilities are of the random access type allowing immediate access to master file information.

FEATURES AND APPLICATIONS

In addition to existing applications best performed by this system, there is a growing number of vital data-processing problems that depend on this type of automation —the UNIVAC 490 Real-Time System—for their effective solution.

The many outstanding features of the UNIVAC 490 Real-Time System are particularly suited to applications in which processing timeliness is vitally important, perishability is a factor, or decision-making is based on data originating simultaneously at separate, remote points. Some of these features, such as solid-state components and microsecond internal computing speeds, represent the latest design advances in the electronic computer field. Other features of the system, such as the ability to communicate with remote locations and to perform both real-time and batch-processing applications, are entirely new developments.

Some of the major features of the UNIVAC 490 Real-Time System are described briefly in the following paragraphs.

Processing Interrupt

An outstanding feature of the UNIVAC 490 Real-Time System is its capacity to process real-time and batch-processing applications concurrently. This impressive data-processing innovation is made possible through a unique feature that permits remote external units to interrupt Computer processing with information of high precedence.

With this feature the maximum processing potential of the system is realized. For example, when once the master data for a particular real-time application is recorded in storage, the Computer can be used to process a "batch" application. Then, whenever transaction data for the real-time problem is entered into a remote external unit, the Computer's batch-processing is interrupted to permit handling the high priority real-time transaction and sending the processed results to the external unit.

Upon completion of the real-time processing, the Computer automatically returns to the batch application. Further, if the real-time information handled during the interrupt bears on the interrupted application, the latter can be updated by the real-time data, thereby assuring that all subsequent processing is up to date.

Solid-State Design

The solid-state components and circuitry of the UNIVAC 490 Real-Time System offer numerous advantages including standardized production of components and the reduction of maintenance procedures to a few relatively simple operations. In addition to ease of production and maintenance, solid-state circuits also impart a high degree of operating reliability to the Computer while reducing the power, cooling, and space requirements of the system.

Computer-to-Computer Configurations

The UNIVAC 490 Real-Time System's ability to coordinate, through communication networks, the activities of several Computers located at various points, allows the user to increase his data-processing system to meet any sudden business expansion.

High-Speed Random Access Storage

To meet the extremely demanding requirements of real-time processing, the UNIVAC 490 Real-Time System is equipped with data-storage facilities (drums, tapes) that are more expansive and versatile than those of most present-day computing systems. The Computer has an internal core memory with a capacity of 32,768 or 16,384 30-bit words.

A "Time Conscious" System

Processing timeliness, an inherent characteristic of real-time processing, requires the system to be extremely "time conscious." Three precision electronic chronometers provide the system with a timing sensitivity unmatched by other computing systems.

Incremental Clock

This built-in clock is used for a wide variety of program-timing purposes. It can be used to log the receipt times of aperiodic real-time input data. Each input message and its receipt time may be recorded together. This clock is also used in connection with the preparation of statistical and analytical reports dealing with the frequency of certain transactions.

Incremental-Interrupt Clock

This "program-set" clock counts up to 32,768 milliseconds. Upon reaching its upper limit, the Incremental-Interrupt Clock unconditionally interrupts the Computer at the end of the instruction being handled, regardless of the type of instruction. The clock count is maintained in core storage. The Incremental-Interrupt Clock is vital to the functioning of a real-time system because one of the primary uses of this clock is timing subroutine operations. If a momentary fault arising from improper programming throws the Computer into a closed loop, or if a fault occurring during the execution of an instruction halts the Computer, the Incremental-Interrupt Clock restarts the Computer by means of an interrupt, thus providing automatic fault recovery. The interrupt can be used to notify maintenance personnel that a closed loop has occurred. If completing an operation takes longer than desired, this clock is also used to interrupt the Computer and thereby allow program attention to be directed to items of more immediate importance.

Day Clock

A feature particularly suited to real-time problems is the 24-hour Day Clock. As an auxiliary device, this electronic clock causes an external interrupt of Computer processing once every minute. Thus, "keeping time" is placed completely under the programmer's direction.

Since program control is shifted once every minute to a set address in the Computer, the Day Clock can be used to initiate a variety of subroutines. For example, by using a compare routine on the address reserved for timing purposes, reports can be generated at any desired time of the day, week or month. These reports could provide up-to-the-minute information and analyses of company status. Error-checking routines, trace routines, output conversions, management report programs, maintenance routines, and memory dumps are some of the many routines which can be initiated by the Day Clock.

High-Internal Computing Speeds

Along with its high-speed random access storage facilities, which allow a high data-transmission rate between the Computer and peripheral units, the UNIVAC 490 Real-Time System also features instruction execution times measured in microseconds. Instruction access and execution time totals 12 microseconds for most instructions.

Equipment Enclosure

The physical arrangement of the Computer, its peripheral units, and the operator's control panel demonstrate the UNIVAC 490 Real-Time System's revolutionary equipment enclosure concept. This modern equipment installation technique affords the operator an unrestricted view of all significant indicators and displays. It also positions the peripheral units within easy reach to allow the operator to attend to them when necessary.

Maintenance personnel, located outside the operator's enclosure, have unrestricted access to all equipment even though it is in operation. Any element of the system may be monitored for maintenance without interfering with operating personnel.

In terms of economy of installation, the equipment enclosure design concept presents the additional advantages of greatly reducing floor-space requirements and eliminating the need for expensive false floors.

Flexible Input-Output Facilities

The UNIVAC 490 Real-Time System can communicate directly with a wide variety of commercially available input-output units and custom-designed data-originating devices. The system's Computer is also capable of communicating directly with other computers. The inherent flexibility of the system's input-output channels enables the UNIVAC 490 Real-Time Computer to perform this diversity of input-output data communicating functions.

A large number of system input-output channels are employed for communications between the Computer and peripheral site units such as high-speed printers, mass-storage units, card readers, and tape-handling units. These same channels can be used to accommodate an almost unlimited number of remote data-originating devices by using a special input-output buffer mode. Two input-output channels are utilized for communications between the UNIVAC 490 Real-Time Computer and other computers.

Automatic Programming

The system features a very flexible compiling system which allows mnemonic expression of computer-oriented instructions. Instructions and addresses to which they refer can be given alpha-numeric names. Program check-out can be accomplished with special aids which include post mortem and register dumping routines. A subroutine mechanism facilities compilation of subroutines into a final program. The compiling system also has a high-level, problem-oriented language in which computer programs are written. This compiler interprets general statements and automatically generates the necessary machine instructions which perform the stated function.

Floating-Point Arithmetic

The UNIVAC 490 Real-Time Computer floating-point format is based on a two-program-word information unit, one mantissa and one characteristic word. The length of the mantissa is 28 bits, and the length of the characteristic is 15 bits, including a sign. It is a three-address system in which four index-registers are used to designate the operand and the function codes. As a complete software system, it includes input-output conversion and function evaluation.

Programming Checks

An important characteristic of the UNIVAC 490 Real-Time System is operating reliability, a mandatory requirement of a system designed for real-time processing. Although the system's reliability is achieved through solid-state components, a number of program checks and error-detection procedures also contribute to the system's highly reliable performance. The system features a routine for automatic restarting after an error is detected and corrected. Because transaction data in real-time applications is usually independent and unrelated to previously entered information, the system provides a routine that preserves the transaction data being processed should an error occur.

SPECIAL PROGRAMMING FEATURES

Special programming features enable the UNIVAC 490 Real-Time System to process virtually any type of commercial or scientific data-processing application. In addition to latest engineering advances incorporated into the logic of the system, a host of special software programming features associated with the system allows the user to cut programming costs to a minimum.

Powerful Instruction Repertoire

The sixty-two function code values in the instruction repertoire can be modified by the system to provide unlimited programming versatility. Much of the real programming power of the computer lies in the unique format of the instruction word. Of the thirty bit positions in the instruction word, nine serve as special purpose designators. When these designators are used in combination with the function codes, the computer can perform more than 25,000 basic programming operations.

Absolute Efficiency

Execution of a stored program of instructions on the Real-Time Computer proceeds in a series of steps. Very often, however, the unique requirements of real-time problems necessitate programming a large number of jump and skip operations. The prevalence of these operations arises because the processing activities of a real-time computer involve serving a large number of remote peripheral units that are constantly demanding computer attention on an immediate or near-immediate basis.

Since skip and jump programming operations are inherent in real-time processing, the internal logic of the UNIVAC 490 Real-Time Computer causes these operations to be handled in a manner approaching absolute efficiency. For example, because the next instruction is in "review" while the current instruction is being executed, a skip instruction brings the computer—at no loss of time—directly to the resumption point without leafing through intervening instructions. Jump instructions are processed by the computer with similar efficiency.

Library of Programmed Routines

A major problem often encountered in the installation of a new type of computing system is the large amount of programming required to put the system in operation. Because the UNIVAC 490 Real-Time System evolved from prototype systems which are now performing successfully at commercial, government, and military installations, an extensive library of proven programming routines, including compiler and assembly systems, is available to the user immediately.

Core Storage Search

From a programming standpoint, a very desirable and practical provision of the UNIVAC 490 Real-Time System is the feature which allows the entire core storage, or selected portions of it, to be searched automatically and rapidly. Programming the search operation requires the use of only two instructions.

Wired Memory

A permanent memory is built into the Computer for program input and automatic error recovery. It consists of sixteen 30-bit words of storage, and is wired to fit the specialized needs of the Computer user. This storage may be accessed by a program, but can only be changed manually.

Application Versatility

Designed primarily for real-time applications, this system can effectively handle numerous commercial and scientific batch-processing applications. Yet it is the real-time function of the system that brings a significant new dimension to electronic data-processing—fingertip access to a powerful computer from many remote points.

The ability to consult vast quantities of updated files in a random manner with immediate program response makes this system ideal whenever an application hinges on exacting time requirements. Thus, whenever time is of great importance as it is where perishability is involved or where customer service must be rapid or where information from several sources must be interrelated for subsequent operations, the UNIVAC 490 Real-Time System can handle the application more effectively than any other system. For example, if a warehouse can make known the availability and quantity of its perishable goods, the points of sale can order their full requirements. In the area of production, to cite another example, work-in-process control can be carefully followed and interim cost figures developed to determine the desirability of alternate methods of production, as well as optimum manufacturing quantities. The UNIVAC 490 Real-Time System, then, has been conceived for use on an unlimited variety of applications. Figure 1–1 shows some real-time data-processing areas listed by industry and application.

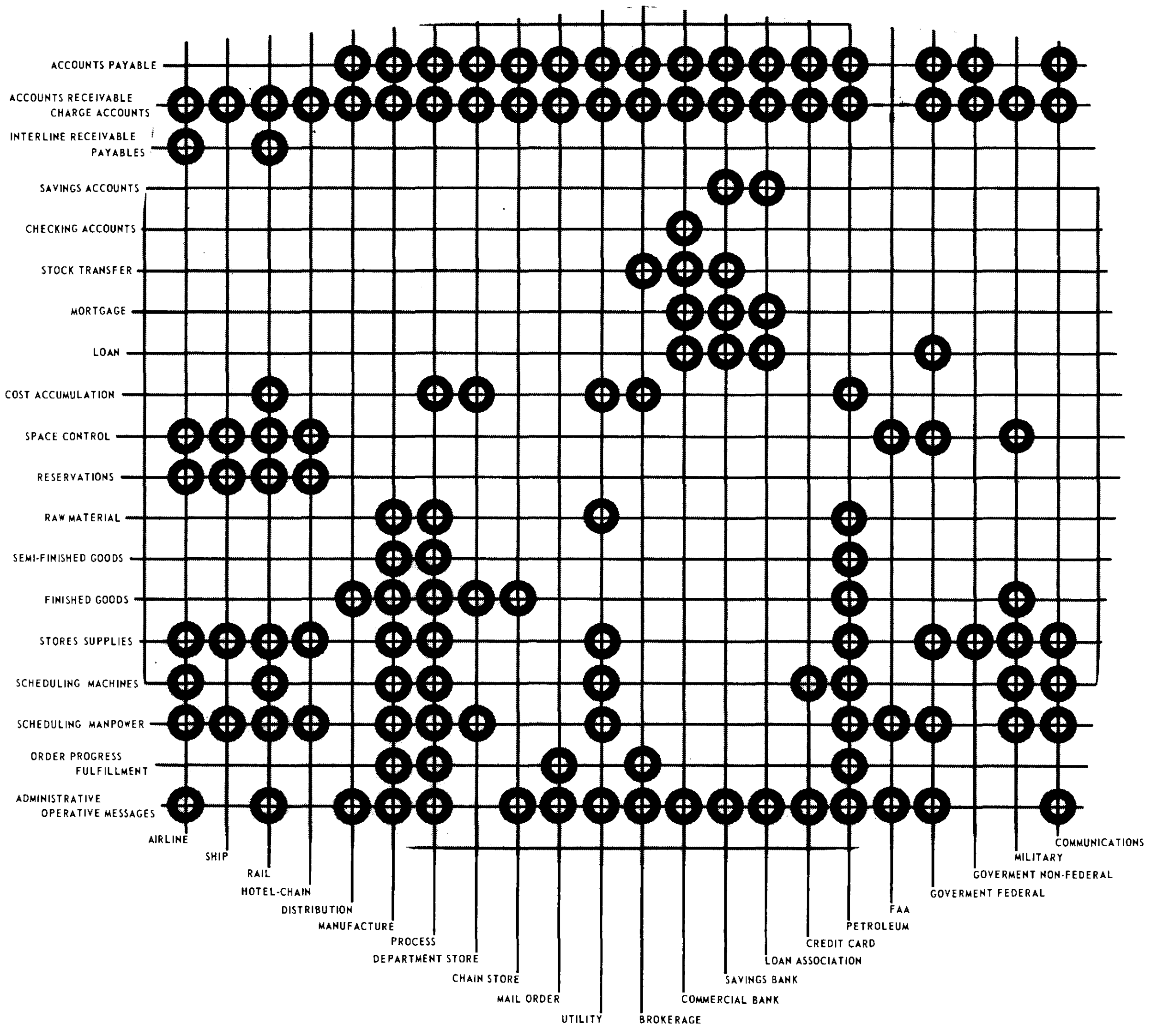


Figure 1-1. Real-Time Data-Processing Applications

2. Real-Time Computer

The Computer in the UNIVAC 490 Real-Time System is a stored program computer designed for processing large quantities of data on a real-time basis. The Computer has large internal magnetic core storage, great programming flexibility, and a versatile input-output section.

The Computer forms the heart of any UNIVAC 490 Real-Time System. Its solid-state arithmetic and logical circuitry perform tens of thousands of processing operations every second, in both batch-processing and real-time modes.

Some outstanding features of the Real-Time Computer are listed below:

- **Access time to all core storage locations of 1.9 microseconds; ability to store and to select information randomly**
- **30-bit word length with a 15-bit half-word option**
- **Repertoire of 62 basic instructions which can be modified to produce over 25,000 different instructions**
- **Single address instructions with provision for address modification**
- **Multiple program capabilities**
- **Ability to perform rapid data exchanges with external equipment without main program attention**
- **Real-time clock for automatically initiating various Computer operations at predetermined times**
- **Parallel one's complement binary notation**

STORAGE SECTION

Internal storage of the Computer consists of banks of ferrite cores. Thousands of these cores can be mounted within a square printed-circuit frame. Each core is capable of assuming either of two stable magnetic states: one represents binary zero; the other, binary one.

At the option of the user, magnetic core storage is available in banks of 16,384 or 32,768 computer words. Access to information in core storage is random since it is independent of the address selected. Words can be inserted into or removed from any address in core storage at a rate of six microseconds per word. Figure 2–1 shows the basic internal data word.

Octal Notation

Although the UNIVAC 490 Real-Time Computer is a binary computer, the problem of converting large decimal numbers into binary notation sometimes becomes cumbersome. For this reason, binary notation is expressed in what is known as octal form. The conversion from binary to octal notation simply involves dividing the binary digits into consecutive sets of three from right to left, and then reading these sets in decimal. For example, a full core storage system requires the use of 32,768 storage addresses. Representation of the upper limit storage address in binary nota-

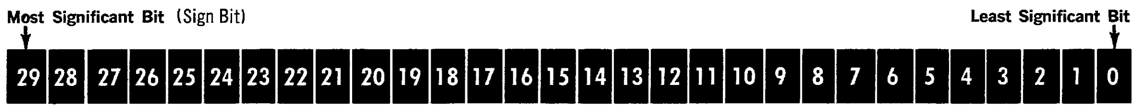


Figure 2-1. Basic Internal Data Word

tion requires the use of 15 bits. This same decimal number, however, can be represented by five octal digits.

Decimal			32,767		
Binary	111	111	111	111	111
Octal	7	7	7	7	7

It should be noted that the working digits in the octal system are 0 through 7. The word octal means eight; therefore, when counting in octal notation, the number after seven is ten. In the Real-Time Computer, the function code values, the operand address, and the operand itself (when the 15-bit option is used) are expressed in octal notation.

CONTROL SECTION

In addition to the magnetic core storage section, the Computer has two other sections, arithmetic and control (Figure 2–2). The control section is responsible for the operations that take place during the sequential execution of instructions. It also coordinates the flow of data between the arithmetic and storage sections.

ARITHMETIC SECTION

The arithmetic section is composed of the circuits and registers used to perform arithmetic and logical operations. These operations are performed in a parallel bi-

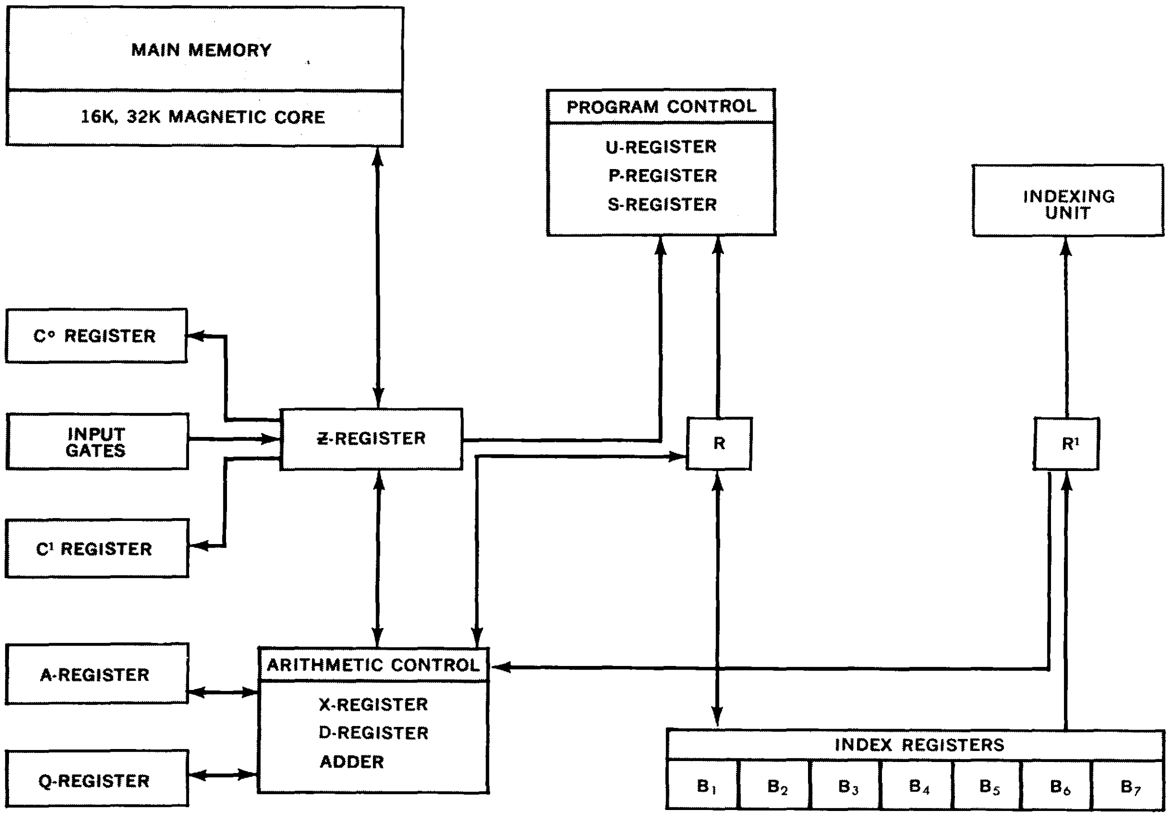


Figure 2-2. Simplified Logical Diagram of the UNIVAC Real-Time Computer

nary mode. The arithmetic mode is one's complement subtractive.

The high internal computing speeds of the Computer allow arithmetic operations to be carried out at microsecond speeds. For example, additions and subtractions are performed in a maximum of 12 microseconds. The ability of the computer to perform arithmetic operations simultaneously with input-output data transfers makes it ideally suited to the processing of real-time problems.

The UNIVAC 490 Real-Time Computer contains a number of registers which holds data during computation. These registers are designated by a letter or letter-numeral combination, and they are interconnected by parallel transmission paths through which information flows during processing.

They fall into two categories: operational and transient. Operational registers contain information from one instruction to another and are referred to in the operational description of each instruction. Transient registers, on the other hand, are temporary storage locations that are always cleared at the end of an instruction.

Arithmetic Registers (Operational Registers)

A-Register

The A-Register or Accumulator is the principal 30-bit arithmetic register. It has adding and shifting properties. In most arithmetic operations, the result is retained in register A for use in later program steps. For example, after addition or subtraction, the sum or difference remains in the accumulator; after multiplication the most significant half of the product is gathered in the accumulator; after division, the remainder is left there.

The contents of register A may be shifted right or left, as described in the instruction repertoire. Left shifts are circular or cyclic, and in a right shift the sign bit is extended by the number of bit positions shifted and the lower order digits are discarded.

Q-Register

The Q-Register is a 30-bit auxiliary arithmetic register. Its principal function is to assist the A-Register in multiply, divide, and logical operations. Register Q has shifting and logical properties, and performs adding or counting functions as well.

The contents of the Q-Register may be shifted right or left, in the same manner as the contents of the A-Register. As shown in Figure 2–2 all communication with the Q-Register is through the X-Register. Logical multiplication is performed on a transmission path between the Q- and X-Registers.

A- and Q-Registers In Combination

Certain instructions shift the contents of the A- and Q-Registers as a single 60-bit register, with the A-Register representing the most significant half of the double-length quantity. To illustrate, the Q-Register holds the multiplier at the beginning of a multiply operation. As the product is formed, by repeated additions and shifts, the multiplier digits are shifted to the right and discarded. In their place, the lower order digits of the double-length product are shifted into the Q-Register from the accumulator.

During a divide operation, a process essentially the reverse of multiplication takes place. The double-length dividend is shifted to the left, and the quotient bits are inserted in the rightmost position of the Q-Register. At the end of the divide sequence, the quotient is assembled in the Q-Register, and the remainder is left in the accumulator.

P-Register

The P-Register (15-bits) is the Program Address Counter. This register holds the address of the next sequential instruction throughout the program. As each program address is transferred from the P-Register to the S-Register, the contents of the P-Register are increased by one. When Jump instructions are executed, the P-Register is cleared and a new program address is entered.

B-Register

The B-Registers (15-bits each) are Address-Modifying Registers generally used for indexing minor loops in a program. The contents of one register may be used to increment the operand address before execution of an instruction. Seven such registers are provided and labeled B^1 through B^7. The B^7 register also serves as a counter in the repeat mode where a selected instruction is executed the number of times specified (covered later in instruction 70).

Transient Registers

The following registers are used in the manipulation of instruction words and data words during the execution of an instruction. These registers are not referenced in the description of the instructions and do not retain information from one operation to the next.

X-Register

The X-Register (30-bits) functions as an arithmetic communication register. It has complementing, but not shifting, properties. The X-Register receives the operand from storage during all arithmetic operations. All communication between the A- and Q-Registers and the rest of the operational registers or the adder output is via the X-Register.

K-Register

The K-Register (6-bits) functions as a shift counter for all arithmetic operations involving shifts. The maximum shift count permitted is 60. Multiply and divide operations are controlled by presetting the K-Register to 30. The K-Register then counts the operational steps.

S-Register

The S-Register (15-bits) holds the storage address during memory references. At the beginning of a storage access period, the address is transferred to the S-Register. The contents of the S-Register are then translated to activate the storage selection system.

Z-Register

The Z-Register (30-bits) serves as an Operand Buffer for storage references. During the read portion of the storage access period, the Z-Register is cleared. The digit-reading amplifiers are then sampled to set the contents of Z corresponding to bits in the storage. During the write portion of the storage access period, the Z-Register controls the inhibit circuits in order to write or restore the disturbed storage register. Input data is gated directly to the Z-Register.

U-Register

The U-Register (30-bits) is the Program Control Register. In other words, it holds the instruction word during the execution of an operation. The operation code and the various execution modifiers are translated from appropriate sections of this register. If an address modification is required before execution, the contents of the appropriate B-Register are added to the contents of the low order 15-bits of the U-Register.

R-Register

The R-Register (15-bits) functions as a communications register for all internal transmissions to the B-Registers.

R^1-Register

The R^1-Register (15-bits) functions as a communication register for all internal transmission from the B-Registers. It holds the incrementing quantity during address modification.

D-Register

The D-Register (30-bits) is the Arithmetic Register which holds the operand, for presentation to the adder, during the execution of arithmetic operations.

C-Register

The C-Registers are communication buffer registers through which computer output data are synchronized. There are two C-Registers, C^0 and C^1. C^0 is used to communicate output data to peripheral devices on 12 different channels. C^1 is used to communicate output data on two different channels to other computers. Input data is gated directly to the Z-Register.

OPERATOR CONSOLE

The UNIVAC 490 Real-Time Computer is equipped with a console that includes controls that allow varied manually governed operations including special modes, Incremental Clock Interrupt Disable Switch, Programmed Jump Switches, and the Wired Memory Switch.

The two-way Wired Memory Switch, located on the operator console, is marked "start" and "neutral."

Computer Control Panel

Registers found on the maintenance panel include the following:

1. C_0-REGISTER	8. B_4-REGISTER
2. C_1-REGISTER	9. B_5-REGISTER
3. Q-REGISTER	10. B_6-REGISTER
4. A-REGISTER	11. B_7-REGISTER
5. B_1-REGISTER	12. U-REGISTER
6. B_2-REGISTER	13. S-REGISTER
7. B_3-REGISTER	14. P-REGISTER

Manual controls are provided on the Computer Control Panel which allow:

1. The execution of consecutive program steps at a low rate.
2. The execution of one consecutive Computer clock phase (¼ of a cycle) for each depression of a switch.
3. The execution of one consecutive program step for each depression of a switch.
4. Operation that is normal except that the Computer does not stop when it executes a programmed stop instruction.
5. The Day Clock to be disconnected.
6. The Increment Clock to be disconnected.
7. The automatic recovery feature to be disconnected.

Because of their use in programming operations, some of the indicators and manual controls listed above are duplicated on the operator's console.

Console Keyboard and Printer

Located at the Computer Console, the Console Keyboard will normally be usable: during program debugging, while making changes to programs, schedules, or tables; when initiating type-outs of interest to the operators or UNIVAC Center supervisory personnel; and in controlling the system. There are no restrictions as to the size of the units of information entered by this device, or of the type-outs, so long as computer formats are used and program provision is made.

Wired Memory

As mentioned earlier the purpose of the wired memory is to provide automatic reading of new programs into the Computer with protection against erasing vital instructions in the wired memory. The wired addresses parallel the first 16 core storage addresses (00–17, octal). Whether the Computer operates with words in wired or core, depends on the position of the 3-way Wired Memory Switch on the Computer Control Panel and the two-way switch on the Operator Console.

The positions of this switch are:

1. Automatic Recovery
2. Neutral
3. Bootstrap

When the Computer is on and the Wired Memory Switch is turned to Bootstrap, the Computer starts the wired program in the wired memory at address 00. During the normal operation of the Computer, any reference to the address between 00 and 17 (octal) refers to the address in the wired memory.

When the switch is in Automatic Recovery, and a fault interrupt occurs, the Computer will perform the program as wired in the Wired Memory starting at address 14. A fault interrupt is caused when an Incremental Clock Interrupt or millisecond time-out or an illegal function (00 or 77) occurs. A millisecond time-out occurs when, for some reason, the Incremental Clock was not updated. All other references to addresses between 00 and 17 (octal) refer to words in core storage.

When the switch is in Neutral, the Computer ignores the wired memory and uses address 00 through 17 (octal) in core storage.

3. System Components and Configurations

The individual components which combine to form a particular UNIVAC Real-Time System configuration vary in number and type according to the application. Each component of a UNIVAC Real-Time System, however, falls into one of the following categories:

1. Remote Inquiry-Answering Devices
2. Communications Equipment
3. Central Site Equipment

Remote inquiry-answering devices situated at many different locations have access to the Computer and other central site equipment through communications equipment. Conversely, the same communications equipment provides the central site units with access to the remote units.

The communications equipment is divided between the remote inquiry-answering devices and the central site equipment. In actual operation, input information is entered at a remote inquiry-answering device which activates communications equipment at the same location. The information is transmitted via communications lines to communications equipment at the central site where it is fed into the Computer for processing. Output information emanating from the Computer is fed through the communications equipment to the proper inquiry-answering device at a remote location.

CENTRAL SITE EQUIPMENT

Central site equipment used with a UNIVAC Real-Time System includes a Real-Time Computer, various peripheral units such as card readers, storage units, and magnetic tape units, and communications equipment. Using master data maintained at the central site, the Computer and its peripheral units process all information received from input-output devices at many remote locations. By means of the communications equipment, input is received by the Computer and results are returned to the inquirer.

The following paragraphs discuss the Real-Time Computer and its peripheral units.

Peripheral Units

The Real-Time Computer can accommodate a large variety of peripheral units. For example, it can handle UNIVAC Solid-State Subsystems equipment such as the High-Speed Printer, the High-Speed Card Reader, and the Punch-Verifier Unit. Uniservo IIA and Uniservo III tape handlers can be connected to the system. In addition, mass-storage devices of the random access type such as flying-head drums, can also be incorporated into a Real-Time System.

Peripheral Systems

The central site peripheral systems are listed as follows:

Flying-Head Drum Subsystem

A Channel Synchronizer
A Drum Control Unit
One to eight Drum Units

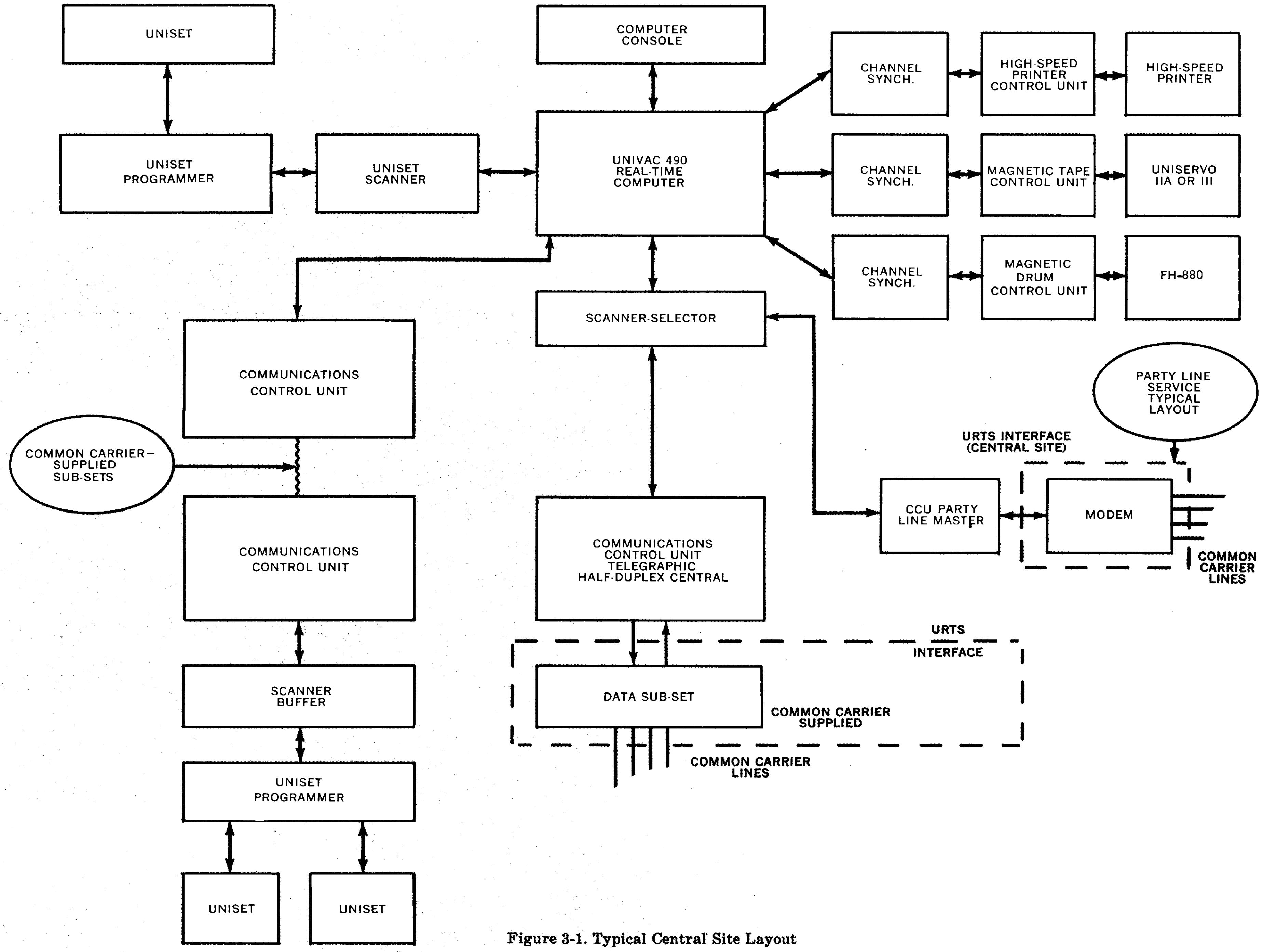

UNISET
COMPUTER CONSOLE
CHANNEL SYNCH.
HIGH-SPEED PRINTER CONTROL UNIT
HIGH-SPEED PRINTER
UNISET PROGRAMMER
UNISET SCANNER
UNIVAC 490 REAL-TIME COMPUTER
CHANNEL SYNCH.
MAGNETIC TAPE CONTROL UNIT
UNISERVO IIA OR III
CHANNEL SYNCH.
MAGNETIC DRUM CONTROL UNIT
FH-880
SCANNER-SELECTOR
COMMUNICATIONS CONTROL UNIT
PARTY LINE SERVICE TYPICAL LAYOUT
URTS INTERFACE (CENTRAL SITE)
COMMON CARRIER-SUPPLIED SUB-SETS
CCU PARTY LINE MASTER
MODEM
COMMON CARRIER LINES
COMMUNICATIONS CONTROL UNIT
COMMUNICATIONS CONTROL UNIT TELEGRAPHIC HALF-DUPLEX CENTRAL
URTS INTERFACE
SCANNER BUFFER
DATA SUB-SET
COMMON CARRIER SUPPLIED
COMMON CARRIER LINES
UNISET PROGRAMMER
UNISET
UNISET

Figure 3-1. Typical Central Site Layout

Magnetic Tape Subsystem

A Channel Synchronizer
A Magnetic Tape Control Unit
One to Twelve Magnetic Tape-Handling Units (Uniservo IIA)

High-Speed Printer Subsystem

A Channel Synchronizer
A High-Speed Printer Control Unit
One or two High-Speed Printers

Card Equipment Subsystem

A Channel Synchronizer
A Card Equipment Control Unit
A High-Speed Reader
A Punch-Verifier Unit.

To communicate with the computer each peripheral subsystem utilizes an input-output channel. Any one of the peripheral systems may be connected to any one of the 12 input-output computer channels. Thus as many as 12 peripheral subsystems in various combinations may be connected to the 12 Computer input-output channels.

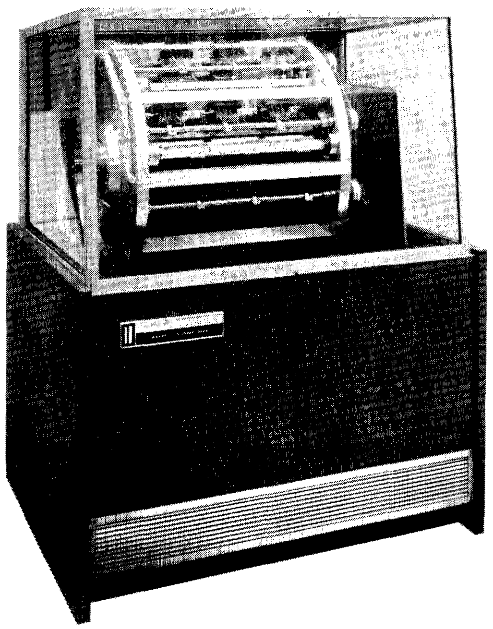
Magnetic Drum Storage

In addition to major features such as large capacity and high-speed random access these mass-storage units employ the flying head (air-floating head) technique which combines aerodynamic and pneumatic principles. The read-write heads float at one-half a thousandth of an inch or less from the oxide-coated surface of the drum, on a boundary layer of air, generated by the rotation of the drum. The read-write heads are suspended in position by the opposing forces of the boundary layer of air and the head-positioning mechanism.

Flying-Head Drum Storage FH-880

In the FH-880 Drum Storage Unit forty head blocks are positioned around the drum. Mounted in each are 22 read-write heads, one for each recording track which revolves beneath the block. The read-write heads record information on the drum surface at a density of 490 bits per inch while it revolves at 1800 revolutions per minute. Recording frequency is approximately 1 megacycle.

There are 128 6-track bands across the drum. Each band can store 6,144 computer words, allowing a total of 786,432 words to be recorded on the drum. Average access time to information stored on the FH 880 drum is one-half drum revolution or 17 milliseconds.



Read-Write Operations

The 30-bit computer words are divided into five 6-bit groups as they are recorded on the surface of the drum. Each 6-bit group is written in parallel followed by the next group and so on until the word is written. Each word is considered one angular address and has a parity bit associated with it.

When performing a read or write operation on the flying-head drum storage, the computer:

1. Sets up a buffer mode
2. Sends a function word to the storage control unit via the channel synchronizer.

The function word contains the starting address and a code specifying which operation is to be performed. The end of the Computer buffer mode will stop the operation.

When a read is to be performed, the storage system locates the starting address and reads the word at that location, and then the input request signal is sent. If an "acknowledge" is received before the beginning of the next word, the information can be read into the Computer as fast as it comes off the drum. This rate can be varied by integral powers of 2, starting from 16.4 microseconds up to a maximum of 262 microseconds. Consecutive words are read into the Computer until the buffer mode ends.

The write operation is similar to the read operation except that the output request signal is transmitted to the Computer immediately upon receipt of the function word. Then the storage system looks for the address and receives a word from the Computer at the same time.

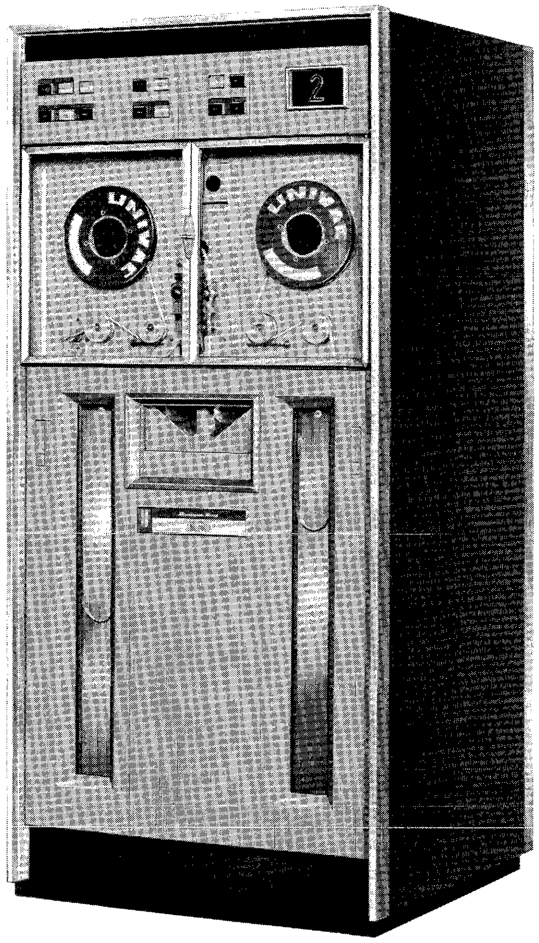
Drum Search

Two types of drum search are available, each using a one-word identifier. One type will locate the matching word and store its address. The other initiates a read operation starting with the word following the matching word.

Overflow words can be inserted at any location preceded by an end of block which is represented by thirty 1-bits. These words will contain an address and an identifying code. When an overflow word is reached during a search, the search will stop at that point and the overflow address is sent to the computer. The search can be reinitiated under program control at the location specified by the overflow address. The End of Block word can also stop a search before a find has been made.

Magnetic Tape Storage

As many as 12 UNISERVO IIA magnetic tape units may operate with a UNIVAC Real-Time Computer through a tape control unit and a channel sychronizer.



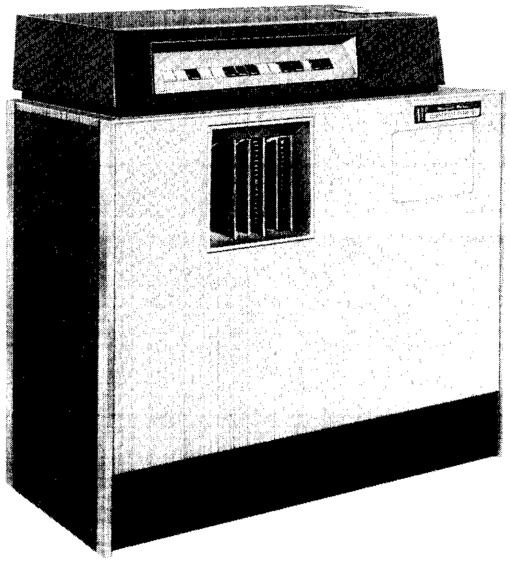
The multiple control units and large storage facilities of the Computer provide for simultaneous tape reading, tape writing, and computation. This means that a 2400-foot reel of tape, containing a data-history file, may be read, updated, and rewritten on an output tape in the time required to read the original file.

The tape units provide fast tape-mounting and ease of operation. A switch on the front panel of each unit permits interchanging of metallic or Mylar* tape. A number of checks is incorporated into the unit to provide and maintain accuracy of information as it is read or recorded.

*Mylar is a registered trademark of E. I. du Pont de Nemours and Company, Inc.

Tape can be moved in a forward or backward direction at a speed of 100 inches per second. Reversal time is 600 milliseconds. Data bits are recorded on the tape in parallel form and characters in serial form at a density of 250 or 125 bits per inch. There are eight recording channels, one is used for parity bits, two for the zone; four, for numerics; and one, for the sprocket pulse. Thus, any 6-bit code plus an odd or even check bit may be employed.

The UNIVAC Real-Time Computer can also accommodate UNIVAC III System magnetic-tape-handling units.

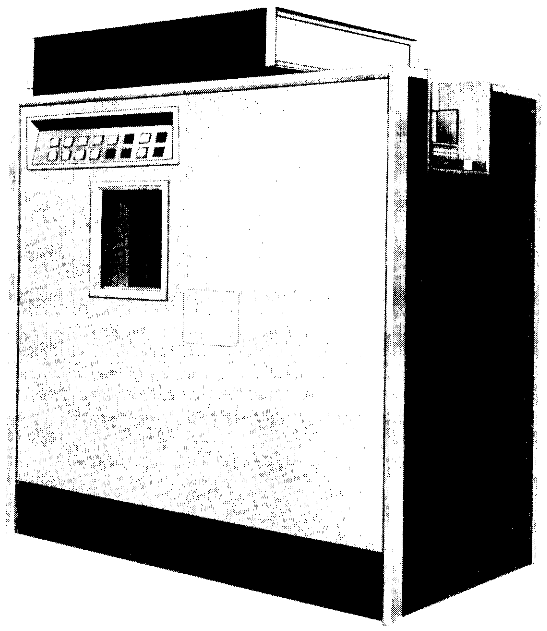


HIGH-SPEED CARD READER

The 80-column card High-Speed Reader is a fast, accurate punched card input unit capable of reading up to 600 cards per minute. The complete accuracy of each read operation performed by the High-Speed Reader is assured through two distinct readings of each 80-column input card.

Data cards stacked in the input magazine are fed into the continuously revolving rollers that transport them through both reading stations, after which they are sent to a stacker. During its course through the transport mechanism, a card conveys its contents to the Computer when it is brush-sensed at the second read station. When once inserted between the rollers, a card moves without interruption until it reaches its stacker. If it were possible to halt the High-Speed Reader instantaneously in the midst of its card-cycling, the following situation would be revealed. First of all, four cards would be committed to the system. The first card in transit from the input magazine, would be near the rollers. Depending on the exact stopping point in a cycle, the next two cards, would be either at or approaching each of the two reading stations. The progress of the fourth card would have been suspended at some point on its way to a stacker. It is this ability to feed and read cards at the same time which enables the High-Speed Reader to reach and sustain its input speed at a maximum rate.

Misfeeds, row misregistration, jams and full stacker are detected; a stacker full signal is provided. Reading at two stations provides full data-checking.



PUNCH-VERIFIER UNIT

The 80-column Punch-Verifier Unit can punch up to 150 cards per minute. The unit consists of an input magazine, a punch station, a post-read station, and two output stackers. The post-read station assures the accuracy of information punched into cards. Verification of punched data is performed at the post-read station on a hole count basis.

A word-for-word check or verification may be easily programmed, eliminating the necessity for a complicated checking procedure. A compact control panel, in open view, advises the operator of all conditions occurring within the unit at any given moment.

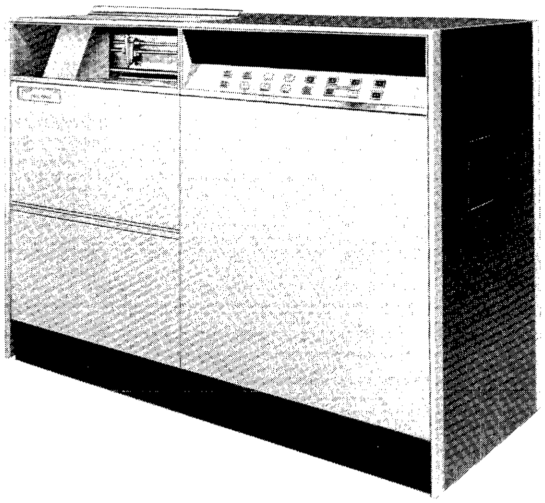
The input magazine, or card-feed bin, holds the stack of cards into which information is to be punched. It has a holding capacity of 600 cards. There is a card-lifting lever which may be depressed to assist in the removal of unprocessed cards. A card weight which fits over the top card in the input stack ensures a positive, uniform feed, down to and including the final card.

During the time that a card is in transit to the punch station, the proper punches for that card can be set up. Electrical signals are used to energize a set of actuator magnets. All of the actuators are contained in a 960 magnet matrix, one for each possible hole position on the card, and the energized actuators correspond to the positions which are to be punched. After the card has been punched, a cam action then opens the card stop and the card moves to the post-read station for verification. From the post-read station, the cards move into the stackers.

Two output stackers, or bins, are contained in a convenient hinged assembly which swings out for maintenance. The bins are designated stacker 1 and stacker 0. Each stacker is capable of holding 850 to 1200 cards. When either stacker is full, microswitches underneath it stop the unit, and an indication is given on the control panel.

THE HIGH-SPEED PRINTER

The High-Speed Printer is capable of printing output information, intermediate results, distributed accumulations and balances, and final reports at a rate of 600 lines per minute.



Basically, the Printer is designed around 128 print wheels which revolve at high speeds. These wheels correspond to the horizontal printing positions across the page. Each print wheel contains 26 alphabetic, 10 numeric, and 15 special characters such as the period, dollar sign, ampersand, colon, and semicolon. Character spacing is ten to the inch across. Line spacing is six to the inch vertically with variable spacing options available to the programmer.

In addition, the paper feed will handle sprocket-fed paper (including card stock, either blank or preprinted) from 4 to 27 inches in over-all width. At least four carbon copies can be made by using paper between 11 and 13.5 pounds in weight. Furthermore, impression control permits variation in the strength of the printing hammer stroke, and fine vertical adjustments of the Printer in operation. The Printer will stop automatically if it detects a low paper supply.

TRANSMISSION AND COMMUNICATIONS

Input-Output Channels

Communication of data between the Real-Time Computer and its central site peripheral units, remote input-output devices, and other computers is achieved through 14 input-output channels. These channels are actually plug-in cables containing various lines which are used to conduct and control information.

Two of these channels are reserved for communication between the Computer and other computers that may be included in a UNIVAC Real-Time System. Depending on the specific requirements of the applications to be performed and on the UNIVAC Real-Time System configuration, the remaining 12 channels may be used as required with remote input-output devices and central site peripheral units.

Data Transfers

Any part of the Computer's internal core storage can be used as an input-output data buffer storage area, with the exception of the few special core storage locations that are reserved for the Incremental Clock and the interrupts. The Computer itself can continue main-program processing without constant interruption by the various input-output units. Information is transferred between the Computer and its external units (remote input-output devices and central site peripheral units) in "blocks" of data. A block of data is a series of words or half-words having consecutive core memory addresses, starting with a program-determined first word and ending with a program-determined last word or an external interrupt.

Buffer Mode

A buffer mode transfer, which occurs independent of main program control, is used to transfer blocks of data between input-output units and core storage. Before execution of a buffer mode transfer of data, the program must perform the following steps:

1. Activate the channel to be used for the information transfer, allowing the Computer to recognize data requests on that channel.
2. Load the channel's index register with the data control word. (The lower and upper halves of the data control word contain the beginning and ending address of the section of core storage involved in the transfer.)
3. Send the proper function word or words to the peripheral unit.

Data is then transferred between the Computer core storage and the peripheral device without main program intervention. When a word is transferred to or from storage, one is added automatically to the lower half of the control word. The data transfer is terminated when the Computer senses that the upper and lower halves of the control word are equal. Steps one and two above are accomplished with one of the Initiate Buffer instructions, and step three is performed by the Enter External Function instruction.

Input-Output Control

Equipment external to the Computer is controlled by function words which activate external function control lines. The function word is translated into a discrete set of actions codes by the external unit. The use of program-generated function words, rather than Computer instructions, in controlling peripheral equipment allows the Computer logic to be independent of the characteristics of input-output units. This permits nearly any type of digital device to be connected to the Computer without modifying computer logic.

Interrupt words are generated by the external units and transmitted to the Computer on the input data lines to inform the Computer program of the operating status of the peripheral unit or its desire to initiate or terminate a transfer. An interrupt word is distinguished from data by activating the external interrupt control lines. Activation of these lines causes the Computer program to jump unconditionally to a storage address associated with the particular input-output channel. From there, the Computer will execute a subroutine which analyzes the interrupt word and prescribes the appropriate action to be taken (Figure 3-2).

External Equipment Requirements

All peripheral units employed in real-time systems must have characteristics suited to both the system and the application. The following capabilities are invariably required of any peripheral unit in a real-time system:

1. Ability to accept, translate, and execute function words.
2. Ability to respond to data-channel input-output acknowledge signals.
3. Ability to produce input-output request signals.

The following capabilities are almost always required of peripheral units in a real-time system:

1. Ability to generate interrupt words.
2. Temporary storage to store as many bits as the peripheral unit transmits in a single transaction.
3. Temporary storage capable of storing the full repertoire of function codes for the specific peripheral unit.

REMOTE INPUT-OUTPUT DEVICES

Because the central site equipment can communicate directly with nearly any type of external digital equipment, remote inquiry answering devices of many different designs can be part of a UNIVAC 490 Real-Time System. Usually, remote inquiry-answering units are especially designed to meet the specific requirements of a particular real-time application. The following paragraphs describe briefly various remote inquiry-answering units now in use at real-time system installations.

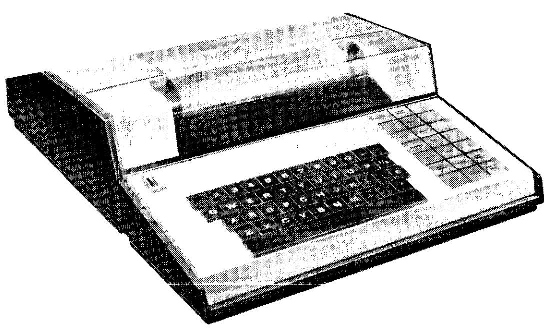
Keyboard Printer

The Keyboard Printer permits keyboard insertion of transaction data and printed page output of computer responses at speeds of 60, 75, or 100 words per minute, depending upon the telegraphic service. The keyboard and printer can be used separately or in combination. Faster printing speeds are available with high-speed telephone communication lines.

The keyboard is the operator's basic means of alpha-numeric input to the system for all information, except that which can be entered via the data keys. The unit employs a full four-bank keyboard with 10 numeric keys, 26 alphabetic keys, 11 special character keys, and a space bar.

NAME	ORIGIN	MEANING	EFFECT
Output Acknowledge	Computer	Computer is transmitting an output word	Initiates cycle whereby output unit accepts and processes output word
Output Data Request	Output Unit	Output unit is ready to receive next output word	Computer forms next output word and sends output Acknowledge
Input Data Request	Input Unit	Input unit is transmitting an input word	Computer accepts input word and sends input Acknowledge
Input Acknowledge	Computer	Computer is receiving the input word	Initiates cycle whereby input unit produces next input word.

Figure 3-2. Input-Output Control Signals

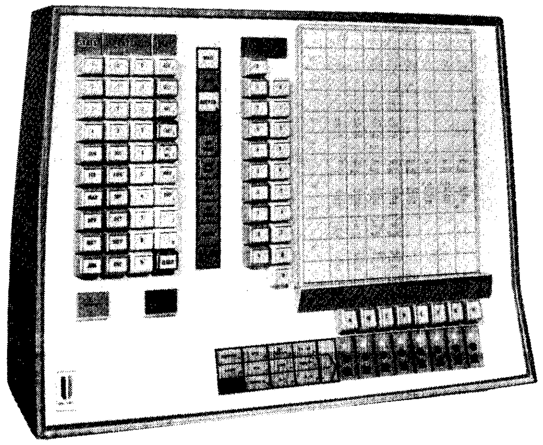


The basic correction facility incorporated in the UNIVAC alpha-numeric equipment operates on a field basis. When an error is detected immediately after it occurs, as in the case of a wrong key being operated, the field-erase key is depressed, resulting in the printing of an error symbol after the faulty entry. The correct entry can then be made. If an error is not discovered until after the transaction has been entered, it can be corrected by means of an appropriate change transaction.

The printing unit prepares a copy of all transaction data as it is typed on the keyboard. Computer responses are also printed by the printing unit.

The Keyboard Printer may be supplied with special features which make it an extremely versatile input-output device. For example, the unit can be used to prepare multilith masters, multiple interleaved carbon sets, horizontal and vertical tabulation, as well as a large variety of printed page requirements. For extremely accurate feeding of forms, the unit is also available with pin-feed platens. It may be used to print payroll checks and a variety of order and invoice forms, and can be programmed to extract selected portions of a transmitted message at various locations.

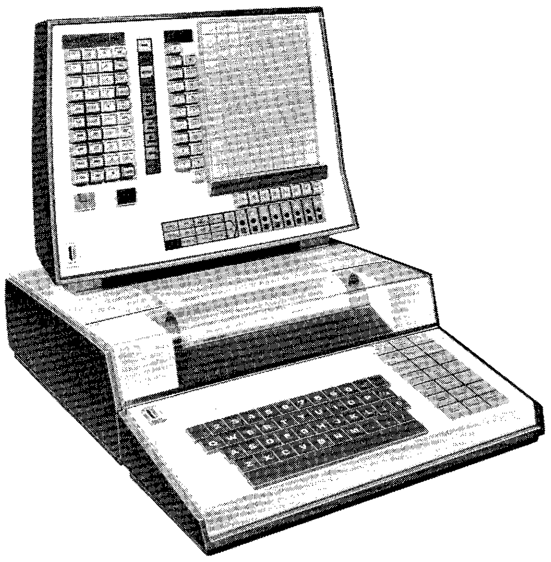
At the option of the user, the Keyboard Printer can be equipped with a format control panel. This auxiliary device can be used to permit greater condensation of data. It also assures that all segments of the data message are transmitted. Housing 21 to 35 pushbutton illuminators, the format control panel is used by the operator to identify specific types of transactions and data fields being entered. By observing which field buttons are illuminated, the operator is informed of transaction data items which are necessary to carry out the indicated transaction.



Uniset Console

The Uniset Console is a keyboard device specifically designed to meet point-of-sale operating requirements of an airline company. Connected on-line with the Computer, this device interrogates the system for flight information by means of its keyboard. Indicator lamps display requested information as well as the Computer's reply. Relevant flight information is stored on a series of transparent cards. When inserted in the Uniset Console by its operator, this information is identified to the Computer.

The Computer identifies each Uniset by a four decimal digit identification number which is read from four manually settable number switches contained within the set's housing. Although developed originally for Airlines Reservations Systems, the Uniset is readily adaptable to various other general applications.



The Uniset

The Uniset is actually a unitive configuration, in a modern modular arrangement, of the combined features of the Keyboard Printer and the Uniset Console.

Functioning as a general-purpose remote input-output device, it allows rapid and accurate interrogation of the Computer. Basically, the Uniset comprises an alphanumeric keyboard, a format control panel, printer card mount, and data keys. The keyboard is the principal medium for inserting transaction data. The unit's four-bank keyboard contains 10 numeric keys, 26 alphabetic keys, and 11 special character keys. Careful attention to color and placement of the various elements permits operation of the Uniset quickly and accurately, with a minimum of effort. Because of the assistance automatically supplied by the equipment during the entry of the data, the complete transaction is carried out smoothly, with a minimum of delay, through the use of the format control keys. The key devices operate with a light touch, yet, like all the Uniset components, are rugged in design to assure reliable operation under the heavy use to which they are exposed.

In addition to printing responses as they are received from the Computer, the printer provides the operator with a printed copy of input transaction data entered on the keyboard. Employing a unique printing mechanism, the unit is quiet in operation and all printed copy can easily be seen by the Uniset operator.

Format Control Panel

The format control panel is a set of 21 to 35 illuminated pushbuttons which are used by the operator to identify specific types of record transactions (for example, new record, reconfirm, cancellation, and so on) and the particular fields being entered (for example, name, phone, remarks, and so on). By observing which field buttons are dimly illuminated, the operator is notified of the items of information required to carry out the transaction he has indicated. Upon depression of one of the dimly illuminated buttons, it "brightens." Upon depression of the next desired dimly illuminated button, that button brightens and the previous brightly illuminated button extinguishes. Thus if an operator is interrupted during a transaction the brightness of illumination of the buttons reminds him of the stage of the entry.

The card mount provides the operator with a convenient and fast method of limiting the inquiry to a specific area or portion of transaction data. The segment data keys consist of three groups of keys which are used for entering specific parts of transaction information.

COMMUNICATIONS EQUIPMENT

In the UNIVAC 490 Real-Time System, remote inquiring-answering devices are connected to the centrally located Computer through communications lines made available by common carriers. Communications equipment located at both the central and the remote sites converts data into a format that can be transmitted over the communication lines. Information arriving at the central site in common carrier form is retranslated by other special equipment into a form usable by the Computer.

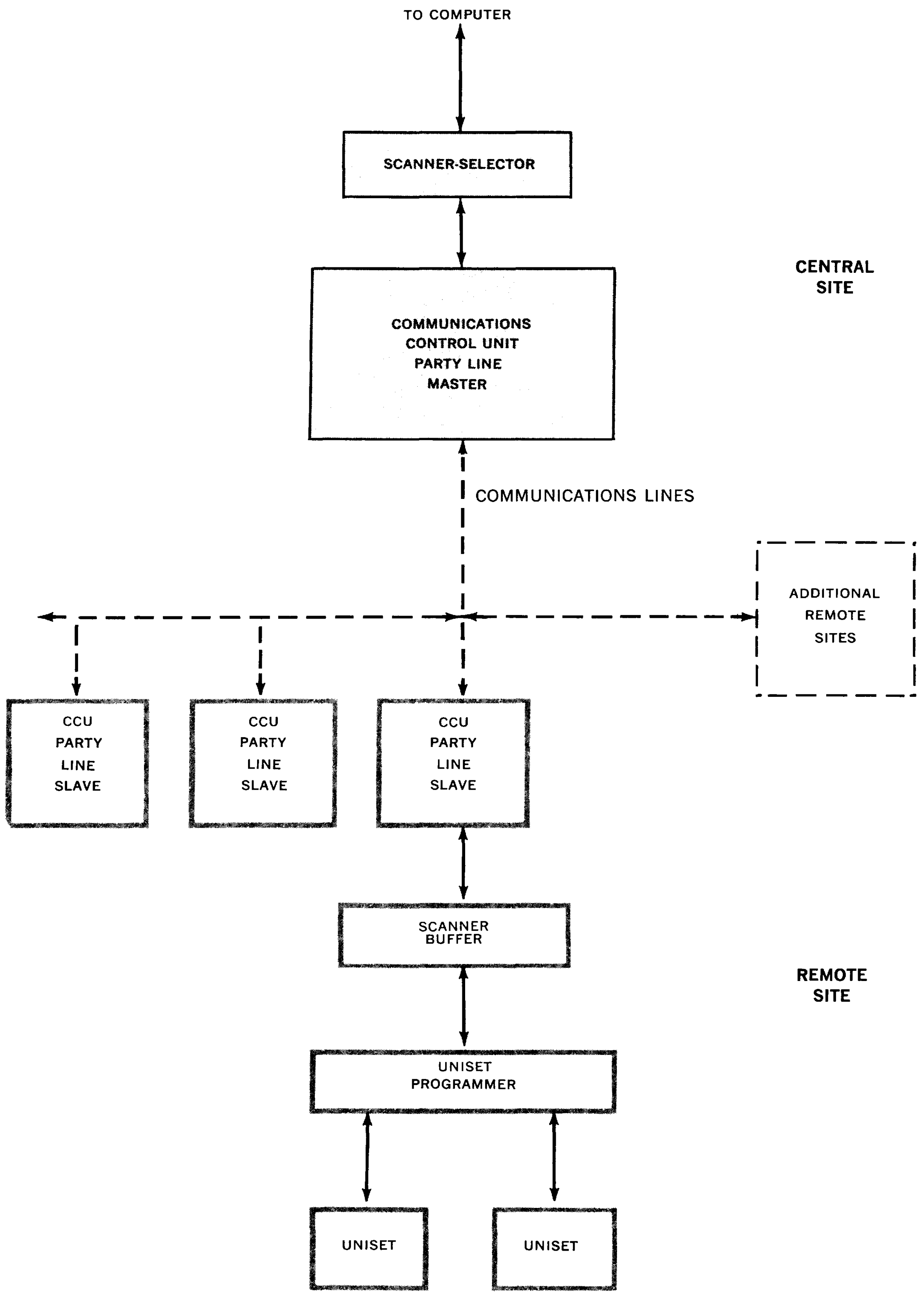


Figure 3-3. Party Line Network

This equipment also controls the various communications circuits to prevent interference and to provide equal availability of circuits to all points having access to the system. Configurations of this equipment vary according to the type of communications lines employed by the system.

COMMUNICATIONS SYSTEMS

Many types of communications systems may be used to transmit information between remote input-output devices and the centrally-located Real-Time Computer. Selection of a system is based upon the application to be performed and the number of remote devices to be used. Configuration of a system varies according to its type. Two of the many communications systems which may be employed are the *party line* and the *line switching network* (Direct Distance Dialing).

PARTY LINE NETWORK

In a party line network (Figure 3-3), remote input-output devices are connected to a Uniset Programmer. This unit may connect, for example, two Keyboards plus Printers to a Scanner-Buffer. The Programmer provides control functions for the two remote devices sharing it.

Scanner-Selector

The Scanner-Selector (S-S) is a device used for connecting up to 16 peripheral units to a single pair of UNIVAC 490 Real-Time Computer input-output channels. It is a simple device in that it does not attempt to directly switch all signal lines on all 32 cables, but instead, switches a single line to each peripheral unit, as an indication that that unit may or may not switch those signal lines it uses. No data storage or buffering is required .in the S-S; each peripheral unit provides its own computer communication capability via a set of common busses. Each S-S is capable of operating with others, for building a cascaded "scanner complex," allowing many more than the basic 16 peripheral units to share a single computer channel pair.

PARTY LINE COMMUNICATIONS SYSTEM

The Party Line Communications System is a digital communications system. It consists of two functional types of digital terminal equipments, master and slave, connected by a communications facility. One master and a number of slave terminals compose a maximum Party Line Communications System. The master terminal is called a Communications Control Unit, Party Line Master (CCU-PLM) and each slave is called a CCU-PLS.

The Party Line Communications System provides bidirectional communication between digital data-processing equipment at the CCU-PLM location and data-handling equipment at each of the CCU-PLS locations. Direct intercommunication between CCU-PLS terminals is not permitted. Only one CCU-PLS may communicate with the CCU-PLM at a given time.

The communications facility is typically a voice-band channel with full-duplex capability. The channel terminations are digital data subsets (modem units). Each slave terminal is electrically bridged across the party line sending pair and receiving pair that terminate at the master station site.

Data transmission between the Central Processor and a remote station is initiated by selective polling (calling) of each CCU-PLS in turn. A polled CCU-PLS communicates with the CCU-PLM as if a direct connection existed between the two with all other CCU-PLS units disconnected. Bidirectional communication continues until the CCU-PLS is told to "hang up" by the data-handling equipment at the remote site. The "Hang Up" signal occurs at the end of a message sent to the Central Processor. The message content indicates to the Central Processor that communication with that station is complete. The Central Processor then instructs the CCU-PLM to poll the next CCU-PLS.

A polled CCU-PLS responds with data when data is made available by a remote data-handling device or with a "No Data" control code. The "No Data" control code causes the CCU-PLM to poll the next CCU-PLS.

Transfer Function

As each Communications Control Unit becomes ready for a one-word buffer transfer, it requests service of the Scanner-Selector (S-S). The S-S sequentially examines the scanner request lines from one of the sixteen stations, and, upon locating a scanner request signal on one of these lines, the scanner transmits a select pulse to the control circuit of that station. The control circuit then initiates the transfer of a data word from the character register to the Computer, or vice versa, depending upon the direction of transfer.

Normally, the scanning circuit is released by terminating the scanner request signal upon the transfer of one Computer word. Transfer of the next word from the same Communications Control Unit is accomplished after one complete cycle of the Scanner-Selector.

The tagging circuit of the Communications Control Unit forms an identifier in the low order 15-bits of the Computer word. The identifier specifies a message buffer area in core storage. The identifier is held until the end of the message while the data character is transferred as the upper 15 bits of the Computer word.

LINE SWITCHING NETWORK (DIRECT DISTANCE DIALING)

The second type of communications line configuration for the Real-Time System is the Line Switching Network (Figure 3-4). This line configuration may use telegraph or voice grade facilities. Line switching service is a network where connections are made only when communications are desired.

COMMUNICATIONS CONTROL UNIT, TELEGRAPHIC HALF-DUPLEX

The Communications Control Unit, Telegraphic Half-Duplex (CCU-THD) is a bidirectional signal converter and adapter for use between a telegraphic channel terminal device and a character input-output register of a computer or remote peripheral device. It serves to relay data from the computer via a telegraphic channel to the remote device, and vice versa. The CCU-THD may utilize Direct Distance Dialing to set up the channel to the proper destination. Operation is half-duplex; therefore, data moves in only one direction at any given time, reversing at some other time.

The CCU-THD handles seven-bit serial asynchronous nine and one-half unit characters at the telegraphic interface, and six-bit parallel characters at the computer or peripheral device interface. It can be adjusted to standard bit rates in the range of 45 to 75 bits per second as well as to high-speed rates in the range of 600 to 750 bits per second if used on voice-band channels. For operation on high-speed channels, polling circuits may be made available to replace the dialing circuits in which case it is called CCU-PHD.

The CCU-THD may use a modified Baudot code of seven bits. The first five bits are standard Baudot code. The sixth bit is used to distinguish letter codes from figure codes. A zero (space) in the sixth bit desig-

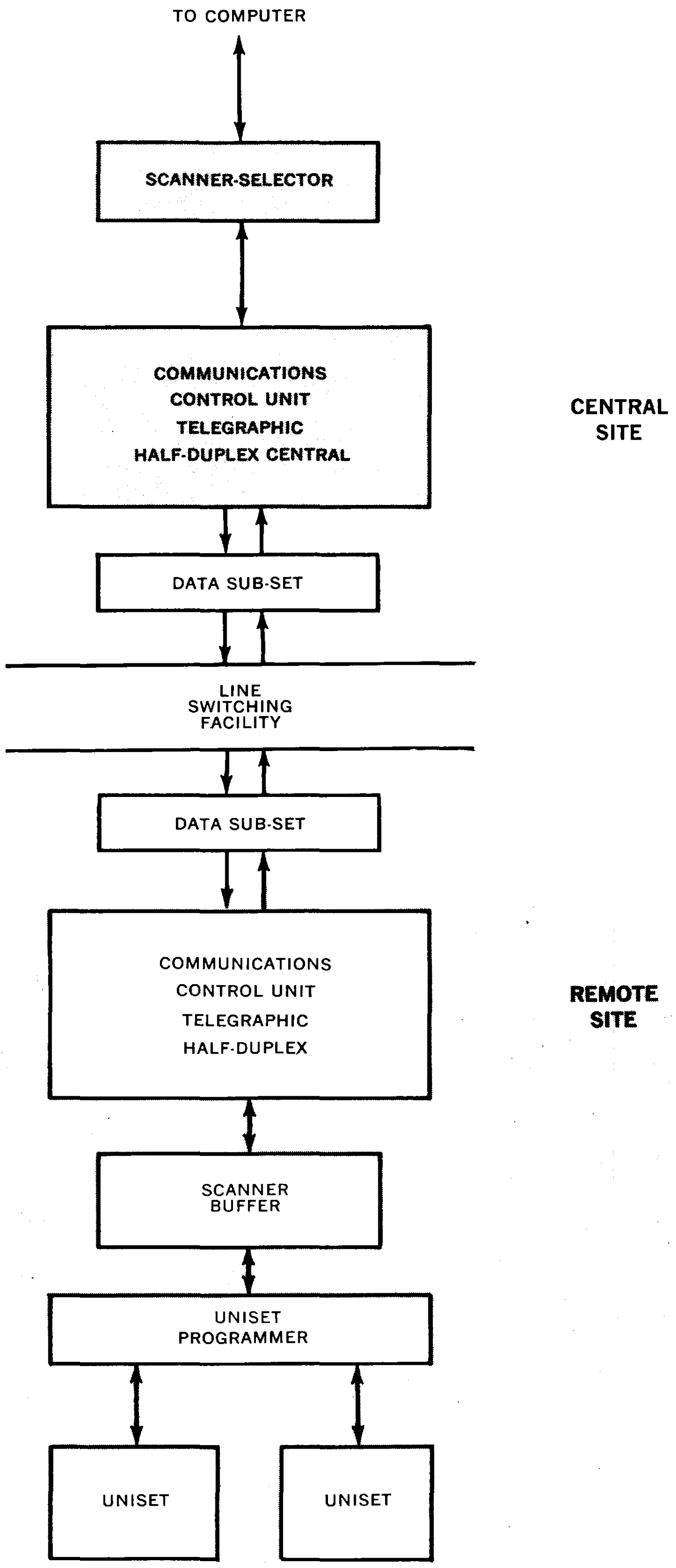


Figure 3-4. Line Switching Network

nates the code as letters, while a one (mark) designates the code as figures. The seventh bit is a parity check bit added by the CCU to data leaving the Computer, and deleted by the CCU from data entering the remote device, and vice versa. The seventh bit is a one or zero as needed to maintain odd parity.

The seventh bit code appears only on the communications channel. The CCU-THD presents an error indicating line and six parallel data lines to the Computer or peripheral device. A one on the error line indicates the concurrent data did not meet the parity check when received. Conversely a zero indicates it did.

The CCU-THD also checks message (longitudinal) parity. The transmitting CCU generates a message parity check code at the end of a message which is compared with the message parity count at the receiving CCU. If they do not compare, a one is presented to the error line.

The CCU-THD contains an interchangeable internal circuit designated a Function Converter which makes its interface compatible with interfaces of a wide variety of data-handling terminal devices.

COMMUNICATIONS EQUIPMENT FOR SPECIAL DEVICES

A Communications Control Unit, Telegraphic Master, Simplex (CCU-TMS) is used for putting teletypewriter devices and data subsets on-line with the real-time system. It is a unidirectional signal converter and adapter for use between a Computer output channel and the Scanner-Selector and telegraphic channel sending terminal. It accepts 6-bit parallel characters from the S-S, converts them to 5-bit code characters, and sends them to a telegraphic transmitting terminal.

Operation of the unit may be adjusted to any standard telegraphic bit rate. The CCU-TMS also generates control pulses for an exchange service telegraphic terminal unit on ten control lines at the rate of ten pulses per second, and in response to dial code 6-bit straight-binary characters received from the Scanner-Selector.

A Communications Control Unit, Telegraphic Slave, Simplex (CCU-TSS) is also used with common carrier teletypewriter equipment. It is a single direction signal converter and adapter used between a telegraphic channel receiving terminal and the Scanner-Selector. It receives 5-bit serial coded characters from the communications terminal and converts them to 6-bit codes for entry into the Scanner-Selector. The CCU-TSS can be adjusted to accept any standard telegraphic character rate.

4. Instructions

INSTRUCTION WORD

As shown in Figure 4–1, each 30-bit instruction word is composed of five designators.

f Designator

The function code designator, f, is a six-bit code that specifies the principal operation—shift, store, add, and so on—to be performed by a program step. Sixty-two function code values constitute the UNIVAC Real-Time System repertoire of instructions.

y Designator

The Y designator is a 15-bit code from which is derived either the address of the operand, or, when the k designator equals 0 or 4, the operand itself.

b Designator

The operand address modification designator, b, is a 3-bit code that governs the first modification of y. This modication involves adding to y the contents of a B-register designated by b.

k Designator

The operand-interpretation designator, k, is usually a 3-bit code that controls the procedure by which the operand is obtained and/or stored. The effect of k is different for each of three instruction categories: read, store, and replace.

The operand interpretation designator also controls data transmissions during store operations.

j Designator

The branch-condition designator, j, is usually a 3-bit code that may be interpreted as a skip or jump-condition designator, a register designator, or a repeat modification designator.

INSTRUCTION CYCLE

The instruction execution cycle begins with a storage access period that transfers the address of the next instruction from register P to register S. Next the 15-bit contents of the storage address register, (register S) are translated, activating the storage selection system, which transmits a 30-bit instruction from memory into register U.

As soon as it enters register U, the instruction word assumes control over the execution of the remaining phases of the program step.

The instruction word function code and various designators are then translated. If address modification is specified, the contents of the designated index register are

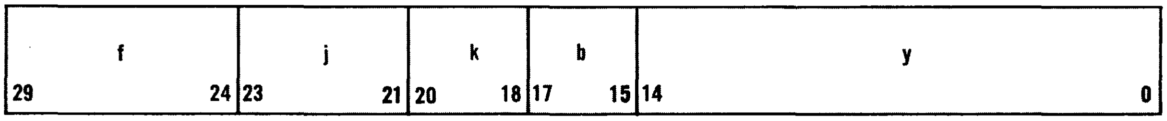


Figure 4-1. Instruction Word

added to the address portion (15 low order bits) of the instruction in register U before execution. Now the control mechanism performs the operations designated by the instruction word; that is, it executes the program step—add, subtract, compare, and so on.

The instruction word remains in register U until it is replaced by the succeeding instruction word at the beginning of the next program step.

INSTRUCTION REPERTOIRE

The programming operations executed by the 62 function code designators are listed below. In describing the operations, the following conventions are used.

( )	the contents of the storage location or register enclosed within the parentheses.
→	the quantity on the left is transferred to the storage location or register on the right.
A	the accumulator, a 30-bit shift register.
Y	the operand address, a 15-bit code that denotes a 30-bit storage location. Sometimes the 15 bits are the operand itself. If so, they are represented
(Y)	by (Y).
AQ	the 60-bit shift register formed by using A and Q together as a single register.
j	the 3-bit branch condition designator.
(A)i (A)f	the initial and final contents of register A, respectively.
Q	the 30-bit quotient register.
Bj	the particular B register designated by j.
Cj*	the particular input-output channel designated by j.
L	the logical product or bit-by-bit product of binary digits.
( )n	the nth bit position of the specified register or address enclosed within the parentheses.
P	the 15-bit program address counter register.
NI	the next instruction to be executed.

SHIFT INSTRUCTIONS

Execution of a shift instruction shifts the word stored in a selected register by (Y) bit positions in a specified direction (right or left). The register selected may be 30-bit register A, 30-bit register Q, or 60-bit register AQ, formed by treating A and Q as a single register. Any right shift of a 30-bit register (A or Q) is open-ended (that is, the rightmost (Y) bits are lost) and provided with sign extension (that is, the leftmost (Y) + 1 bits become duplicates of the sign bit). All other shifts are circular; that is, the (Y) bits that are shifted out of one end of the register appear in sequence at the opposite end of the register (no bits are lost). All shift instructions are Read Class instructions. The shift instructions are as follows:

01 Shift Q Right

This instruction shifts (Q) to the right (Y) bit positions. The higher-order bits are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of (Y) are recognized for this instruction and the maximum shift count permitted is 59. The higher-order bits of (Y) are ignored.

02 Shift A Right

This instruction shifts (A) to the right (Y) bit positions. The higher-order bits are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of (Y) are recognized for this instruction and the maximum shift count permitted is 59. The higher-order bits of (Y) are ignored.

03 Shift AQ Right

This instruction shifts (A) and (Q) as one 60-bit register. The shift is to the right (Y) bit positions with the lower-order bits of (A) shifting into the higher-order bit positions of Q. The higher-order bits of (A) are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of (Y) are recognized for this instruction and the maximum shift count permitted is 59. The higher-order bits of (Y) are ignored.

05 Shift Q Left

This instruction shifts (Q) circularly to the left (Y) bit positions. The lower-order bits are replaced with the higher-order bits as the word is shifted. Only the lower-order six bits of (Y) are recognized for this instruction and the maximum shift count permitted is 59. The higher-order bits of (Y) are ignored.

06 Shift A Left

This instruction shifts (A) circularly to the left (Y) bit positions. The lower-order bits are replaced with the higher-order bits as the word is shifted. Only the lower-order six bits of (Y) are recognized for this instruction and the maximum shift count permitted is 59. The higher-order bits of (Y) are ignored.

07 Shift AQ Left

This instruction shifts (A) and (Q) as one 60-bit register. The shift is circular to the left (Y) bit positions. The lower-bits of (A) are replaced with the higher bits of (Q), and the lower bits of (Q) are replaced with the higher bits of (A). Only the lower-order six bits of (Y) are recognized for this instruction and the maximum shift count permitted is 59. The higher-order bits of (Y) are ignored.

SIMPLE READ INSTRUCTIONS

Execution of a simple read instruction transfers (Y) to a selected register. The selected register may be Q, A, B^j, or channel j. The simple read instructions are as follows:

10 Enter Q-Register

Clear the Q-Register. Then transmit (Y) to Q.

11 Enter A-Register

Clear the A-Register. Then transmit (Y) to A.

12 Enter B-Register

Clear the contents of B-Register j. Then transmit the 15-bits of (Y) as specified by k for read class instructions to B-Register. The Branch Condition Designator, j, is used to specify the selected B-Register for this instruction and is not available for its normal function.

13 Enter External Function

Clear the contents of the C^0 register. Then transmit (Y) to the C^0 register and send an External Function Control signal on external function line C_{j*}. Channels 0 and 1 do not have external function control lines. The k designator equals 3 only.

If $j^* = 0, 1$, the instruction is a conditional skip on an external signal line associated with the specified channel. The channel designator, j*, is used to specify the selected channel. The two bits of k* are interpreted as $k = 0$ or $k \neq 0$. A $k \neq 0$ must be used for $j^* \neq 0, 1$. A $k = 0$ must be used for $j^* = 0, 1$. If $k \neq 0$ the (Y) is used unmodified.

STORE INSTRUCTIONS

Execution of a store instruction stores a selected word at address Y. The selected word may be (Q), (A), (C^{j*}), (A) + (Q), (A) − (Q), or L (A)• (Q). The store instructions are as follows:

14 Store Q Register

Store (Q) at the storage address Y as directed by the operand interpretation designator k for store class instructions. (See Table 3.)

15 Store A Register

Store (A) at storage address Y as directed by the operand interpretation designator k for store class instructions. (See Table 3.)

16 Store B Register

Store the contents of the selected B-Register at storage address Y as directed by the operand interpretation designator k for store class instructions. (See Table 3.) The Branch condition designator, j, is used to specify the selected B-Register for this instruction and is not available for its normal function.

17 Store Input Channel

Store, in address Y, the unmodified content of the channel specified by the channel designator, j*. An Input Acknowledge is sent on the specified channel. The k designator equals 3 only.

32 Add Q and Store

Add (Q) to the previous content of A. Then store (A) at storage address Y as directed by the operand interpretation designator k for store instructions. (See Table 3.)

33 Subtract Q and Store

Subtract (Q) from the previous content of the A-Register. Then store (A) at storage address Y as directed by the operand interpretation designator, k, for store class instructions. (See Table 3.)

47 Store Logical Product

Form in the X-Register the bit-by-bit product of (A) and (Q). Then store (X) at storage address Y as directed by the operand interpretation designator, k, for store class instructions. (See Table 3.)

ARITHMETIC READ INSTRUCTIONS

Execution of an arithmetic read instruction combines (Y) with the previous content of a selected register to form a sum, difference, quotient, or product. The arithmetic read instructions are as follows:

20 Add

Add (Y) to the previous content of the A-Register.

21 Subtract

Subtract (Y) from the previous content of the A-Register.

22 Multiply

Multiply (Q) times (Y) leaving the double-length product in the AQ. If the factors are considered as integers, the product is an integer in AQ.

23 Divide

Divide (AQ) by (Y) leaving the quotient in the Q-Register and the remainder in the A-Register. The remainder bears the same sign as the quotient.

26 Q Add

Shift (A) and (Q) left 30 places as a single 60-bit register. Then add (Y) to (A). Shift (A) and (Q) left 30 places as a single 60-bit register. The content of A is undisturbed by this instruction. The branch condition designator, j, has special meaning in this instruction. (See Table 1.)

27 Q Subtract

Shift (A) and (Q) left 30 places as a single 60-bit register. Then subtract (Y) and (A), shift (A) and (Q) left 30 places as a single 60-bit register. The content of A is undisturbed by this instruction. The branch condition designator, j, has special meaning in this instruction. (See Table 1.)

30 Load A Add Q

Clear A. Then add (Q) to A. The (Y) are then added to (A).

31 Load A Subtract Q

Clear A. Then subtract (Q) from (A). Finally add (Y) to (A).

40 Enter Logical Product

After clearing the A-Register, form the bit-by-bit product of (Y) and (Q) in the X-Register. Then add (X) to (A). The branch condition designator, j, is interpreted in a special way for this instruction for values of $j = 2, 3$. If $j = 2$, skip the next instruction if the parity of the final (A) is even. If $j = 3$, skip the next instruction if the parity of the final (A) is odd.

41 Add Logical Product

Form in the X-Register the bit-by-bit product of (Y) and (Q). Then add (X) to the previous content of A.

42 Subtract Logical Product

Form in the X-Register the bit-by-bit product of (Y) and (Q). Then subtract (X) from the previous content of A.

COMPARISON INSTRUCTIONS

Execution of a comparison instruction involves words in Y, A, and Q, but produces no net change in these registers. The function of a comparison step is to determine whether a "skip" condition obtains. (This point is explained further under "Branch-Condition Designator" below.) The comparison instructions are as follows:

04 Compare

This instruction compares the signed value of (Y) with the signed value of (A) and/ or (Q). It does not alter either (A) or (Q).

The branch condition designator, j, is interpreted in a special way for this instruction. (See Table 1.)

43 Masked Comparison

Form in the X-Register the bit-by-bit product of (Y) and (Q). Then subtract (X) from the previous content of A. Then perform the branch point evaluation of skip of the next sequential instruction as directed by the branch condition designator, j. (See Table 3.) Then add (X) to A. This instruction results in no net change in the content of any operational register.

SELECTIVE INSTRUCTIONS

Execution of a selective instruction alters selected bits of (A) without affecting the unselected bits of (A). The bits of (A) to be altered are determined by a 30-bit selection word; each bit of (A) is subject to alteration if and only if the corresponding bit of the selection word is a "1". The selection word may be either (Y) or (Q); the alteration that is applied to each selected bit may be one of four types:

a. replacement by "1"
b. complementation
c. replacement by "0"
d. replacement by the corresponding bit of (Y) • (Q)

The selective instructions are Read Class instructions and are described below:

50 Selective Set

Set each of the bits of (A) to one where there is a corresponding 1-bit in (Y); the remaining bits of (A) are unaltered.

51 Selective Complement

Complement each of the bits of (A) where there is a corresponding 1-bit in (Y); the remaining bits of (A) are unaltered.

52 Selective Clear

Clear each of the bits of (A) where there is a corresponding 1-bit in (Y); the remaining bits of (A) are unaltered.

53 Substitute

Clear each of the bits of (A) where there is a corresponding 1-bit in (Q); the remaining bits of (A) are unaltered. Then form in the X-Register the bit-by-bit product of (Y) and (Q). Then set each of the bits of (A) to one where there is a corresponding 1-bit in (X); the remaining bits of (A) are unaltered. This instruction has the effect of replacing bits of (A) with bits of (Y) corresponding to ones in (Q).

REPLACE INSTRUCTIONS

Execution of a replace instruction alters (Y). The alteration consists in combining (Y) with the contents of one or more other registers and storing the result at address Y. The replace instructions are as follows:

24 Add Replace

Add (Y) to the previous content of A. Then store (A) at storage address Y.

25 Subtract Replace

Subtract (Y) from the previous contents of A. Then store (A) at storage address Y.

34 Replace Add Q

Clear A. Then add (Q) to (A). Finally add (Y) to (A) and store (A) at storage address Y.

35 Replace Subtract Q

Clear A. Then subtract (Q) from (A). Finally add (Y) to (A) and store (A) at storage address Y.

36 Replace Add One

Clear A. Then add one to (A). Finally add (Y) to (A) and store (A) at storage address Y.

37 Replace Subtract One

Clear A. Then add minus one to (A). Finally, add (Y) to (A) and store (A) at storage address Y.

44 Replace Logical Product

Clear A. Then form in the X-Register the bit-by-bit product of (Y) and (Q). Finally add (X) to (A) and store (A) at storage address Y. The branch condition designator, j, is interpreted in a special way for this instruction for values j = 2, 3. If j = 2, skip the next instruction if parity of final (A) is even. If j = 3, skip the next instruction if the parity of final (A) is odd.

45 Replace Add Logical Product

Form in the X-Register the bit-by-bit product of (Y) and (Q). Then add (X) to the previous content of A and store (A) at storage address Y.

46 Replace Subtract Logical Product

Form in the X-Register the bit-by-bit product of (Y) and (Q). Then subtract (X) from the previous content of A and store (A) at storage address Y.

54 Replace Selective Set

Set each of the bits of (A) to one where there is a corresponding 1-bit in (Y) and the remaining bits of (A) are unaltered. Then store (A) at storage address Y.

55 Replace Selective Complement

Complement each of the bits of (A) where there is a corresponding 1-bit in (Y); the remaining bits of (A) are unaltered. Then store (A) at storage address Y.

56 Replace Selective Clear

Clear each of the bits of (A) where there is a corresponding 1-bit in (Y); the remaining bits of (A) are unaltered. Then store (A) at storage address Y.

57 Replace Substitute

Clear each of the bits of (A) where there is a corresponding 1-bit in (Q); the remaining bits of (A) are unaltered. Then form in the X-Register the bit-by-bit product of (Y) and (Q). Next set each of the bits of (A) to one where there is a corresponding 1-bit in X; the remaining bits of (A) are unaltered. Finally, store (A) at storage address Y.

JUMP INSTRUCTIONS

The Computer ordinarily executes instructions taken from consecutive memory addresses. It can, however, be made to branch to Y, an an out-of-sequence address determined by other designators in the instruction word. This operation is called a jump. A jump diverts computer operation from the main program to a subroutine (or an isolated main-program segment) that begins with execution of the instruction at the out-of-sequence address. A return jump is a special kind of jump that performs the additional function of storing the address of the next main-program instruction (the instruction that would next have been executed if the jump had not occurred) at a reference location in the portion of memory allotted to the subroutine. When a return jump is used, the subroutine that it initiates ordinarily concludes with a jump to the address stored in the reference location. This concluding jump causes the main-program operation to be resumed at the point at which it was interrupted.

Execution of a jump instruction (1) determines whether a jump condition obtains, and (2) performs the indicated jump if the jump condition is satisfied. Jump instructions differ with respect to the conditions on which performance of the jump depends. The jump instructions are as follows:

60 Arithmetic Jump

Execution of this instruction will cause the next program instruction to be taken from (Y) under certain conditions of either the A- or Q-register contents. The branch condition designator, j, is interpreted in a special way for this instruction. (See Table 1.)

61 Manual Jump

Execution of this instruction will cause the next program instruction to be taken from (Y) under certain conditions of manual Jump Key selection. Program execution may be stopped by certain Stop Key selections. The branch condition designator, j, is interpreted in a special way for this instruction. (See Table 2.)

62 Input Active Jump

This instruction clears the program address register P, and enters a new program address in P for certain conditions of the input active designator. The channel designator, j*, is interpreted to designate which channel status will be examined. If the input active designator is in the one state, the jump condition is satisfied. The two bits of k* are interpreted as k = 0, 1, 2, 3.

63 Output Active Jump

This instruction clears the program address register P, and enters a new program address in P for certain conditions of the output action designator. The channel designator, j*, is interpreted to designate which channel status will be examined. If the output active designator is in the one state, the jump condition is satisfied. The two bits of k* are interpreted as k = 0, 1, 2, 3.

64 Arithmetic Return Jump

This instruction executes a return jump sequence for certain conditions of either the A- or Q-register content. The branch condition designator is interpreted in a special way for this instruction. (See Table 2.) If the jump condition is satisfied, the address of the return jump instruction plus one is stored in the address portion of (Y) and a program jump is executed to the instruction at address Y + 1.

65 Manual Return Jump

This instruction executes a return jump sequence for conditions determined by manual key selections. The branch condition designator, j, is interpreted in a special way for this instruction. (See Table 2.) If the jump condition is satisfied, the address of the return jump instruction plus one is stored in the address portion of (Y) and a program jump is executed to the instruction at address Y + 1.

66 Deactivate Input Buffer

This instruction clears the bits specified by the channel designator, j*, in the input active designator and the input monitor designator.

67 Deactivate Output Buffer

This instruction clears the bits in the output active designator and the output monitor designator specified by the channel designator, j*.

SPECIAL PROGRAM-MODIFYING INSTRUCTIONS

A program-modifying instruction is one whose execution alters the normal computer procedure of executing instructions from consecutive memory addresses. In addition to the standard jump instructions described above and the skip option described under "Branch-Condition Designator" below, these three special program-modifying instructions are provided: 'initiate repeat', 'index skip', and 'index jump'.

Execution of an 'initiate repeat' instruction causes the next successive instruction to be executed (Y) times. The repeated instruction may be modified for each repetition as indicated in the tabulation below (where j is part of the 'initiate repeat' instruction while y and b are parts of the repeated instruction itself).

Execution of an 'index skip' instruction can cause the computer to perform a skip; that is, to leave the next sequential instruction unexecuted and execute its successor instead.

Execution of an 'index jump' instruction can cause the computer to perform a jump.

The special program modifying instructions are as follows:

70 Initiate Repeat

Initiate a repeat mode of operation which will cause the next sequential instruction to be executed (Y) times. The remaining executions are maintained in B-Register 7. The branch condition designator, j, is interpreted in a special way for this instruction. (See Table 2.)

71 Index Skip

If the content of the B-Register, specified by the branch condition designator, j, is equal to (Y), skip the next instruction and clear B-Register j. If the content of B-Register j is not equal to (Y), add one to the content of B-Register j and proceed to the next sequential instruction.

72 Index Jump

If the content of the B-Register, specified by the branch condition designator, j, is non zero, execute a jump in program address to address (Y) and reduce the content of B-Register j by one. If the content of B-Register j is zero proceed to the next sequential instruction.

73 Initiate Input Buffer

If $k^* = 3$, store (Y), the input control word, in storage location 00100 + j^*. If $k^* = 1$, store the lower 15-bits of (Y) in the lower-order half of address 00100 + j^*. Then set the input active designator bit j^*. $k = 2$ is not permitted.

74 Initiate Output Buffer

If $k^* = 3$ store (Y) the output control word in storage location 00120 + j^*. If $k^* = 1$ store the lower 15-bits of (Y) in the lower-order half of the address 00120 + j^*. Then set the input active designator bit j^*. $k = 2$ is not permitted.

75 Initiate Input Buffer with Monitor

If $k^* = 3$, store (Y), the control word, in storage location 00100 + j^*. If $k^* = 0$, store Y in the lower half of storage location 00100 + j^*. If $k^* = 1$ store the lower-order 15-bits of (Y) in the lower-order half of address 00100 + j^*. Then set the j^* bit of both the input active designator and the input monitor designator. $k = 2$ is not permitted. A program interrupt to the instruction stored at address 00040 + j^* will occur at the termination of the input buffer.

76 Initiate Output Buffer with Monitor

If $k^* = 3$, store (Y), the control word, in storage location 00120 + j^*. If $k^* = 0$, store Y in the lower-half of storage location 00120 + j^*. If $k^* = 1$ store in the lower-order half of address 00120 + j^*. Then set the j^* bit of both the output active designator and the output monitor designator. $k = 2$ is not permitted. A program interrupt to the instruction stored at address 00060 + j^* will occur at the termination of the output buffer.

Branch-Condition Designator

The branch-condition designator, j, is a three-bit code that may be interpreted as (1) a skip or jump-condition designator, (2) a register designator, or (3) a repeat-modification designator.

Skip and Jump Interpretations of j

The most common use of j is to designate conditions which, if satisfied, will cause the next instruction to be skipped. The normal skip conditions are listed below:

$j = 0$	No skip
$j = 1$	Unconditional skip
$j = 2$	Skip if (Q) is positive
$j = 3$	Skip if (Q) is negative
$j = 4$	Skip if (A) is zero
$j = 5$	Skip if (A) is not zero
$j = 6$	Skip if (A) is positive
$j = 7$	Skip if (A) is negative

The common skip interpretation of j, as defined by the above list, applies to instructions in which f = 01-03, 05-11, 14, 15, 20-25, or 30-37.

Special skip interpretations apply to 'compare', 'Q add', and 'Q subtract' instructions (f = 04, 26, 27 respectively). These interpretations are defined by the list that follows:

	f = 04	**f = 26, 27**
$j = 0$	No skip	No skip
$j = 1$	Unconditional skip	Unconditional skip
$j = 2$	Skip if (Y) $\leqq$ (Q)	Skip if (A) is positive
$j = 3$	Skip if (Y) $>$ (Q)	Skip if (A) is negative
$j = 4$	Skip if (Q) $\geqq$ (Y) and (Y) $>$ (A)	Skip if (Q) is zero
$j = 5$	Skip if (Y) $>$ (Q) or (Y) $\leqq$ (A)	Skip if (Q) is not zero
$j = 6$	Skip if (Y) $\leqq$ (A)	Skip if (Q) is positive
$j = 7$	Skip if (Y) $>$ (A)	Skip if (Q) is negative

In jump instructions (f = 60-67), j designates conditions that determine whether or not the indicated jumps are to be performed. Interpretations for j in jump instructions are defined under "Jump Instructions" above.

(The computer treats zero as a signed number that is usually, but not necessarily, positive. This fact can affect some of the interpretations of j.)

*Register-Specification Interpretation of j or j**

Input and output jump instructions (f = 62, 63, 66, 67), j* does not directly specify a jump condition; instead, it designates the channel whose status (sampled or unsampled) determines the performance of the jump. Similarly, in all other instructions that involve a program-selected channel (f = 13, 17, 73-76), j* designates the selected channel. Similarly, in all instructions that involve a program-selected B-Register (f = 12, 16, 71, 72), j designates the selected register.

Repeat-Modification Interpretation

In 'repeat' instructions (f = 70), j governs modification of the repeated instruction. This interpretation of j is defined under "Special Program-Modifying Instructions" above.

Operand Address Designator

As mentioned earlier, when k=0 or 4, the 15-bit y designator is subjected to modification and becomes the operand which is represented by (Y). The role of the operand in the execution of each instruction depends on f and is explained for each f under "Function-Code Designator" above. The manner in which the operand is developed from y depends on other designators and is explained in the paragraphs that follow.

Address-Modification Designator

The address-modification designator, b, is a three-bit code that governs the first modification to which y is subjected. This modification consists of adding to y the contents of a B-Register designated by b. That is, y is modified to become y + (B^b). No B^0 register exists; (B^0) is interpreted as zero so that y is not modified if b = 0.

Operand-Interpretation Designator

The operand-interpretation designator, k, is a three-bit code that governs, first, the process whereby the operand is derived from y + (B^b), and secondly, data transmissions in store operations. The effect of k is different for each of three instruction categories: read, store, and replace.

Interpretation of k in Read Instructions

In read instructions (f = 01-13, 20-23, 26-31, 40-43, 50-53, 60-76), the transfer of the operand (Y) from the X-Register to either the A, Q, B or C Registers, is determined by the particular function used. The derivation of the operand (Y) in X is determined by k in accord with the list that follows. The list shows the contents of the upper and lower 15 positions of X for each value of k.

k	**$(X)_U$**	**$(X)_L$**
0	Zero	$y+(B^b)$
1	Zero	$(y+(B^b))_L$
2	Zero	$(y+(B^b))_U$
3	$(y+(B^b))_U$	$(y+(B^b))_L$
4	es	$y+(B^b)$
5	es	$(y+(B^b))_L$
6	es	$(y+(B^b))_U$
7	$(A)_U$	$(A)_L$

In the list above, "es" for ("extended sign") indicates that all bits of $(X)_U$ are equal to the highest order bit of $(X)_L$. This list also shows that y + (B^b) may, (1) become part of the operand (k = 0,4), (2) specify the address of part of the operand (k = 1,2,3,5,6) or (k = 7) be replaced by (A). The value k = 7 is excluded for f = 22,23,26-31,52,53.

Interpretation of k in Store Instructions

In store instructions (f = 14-17,32-33,47) the contents of the A, Q, B or C Registers, as determined by the particular function used, is transferred to the D Register. The derivation of the operand (Y) in D is determined by k in accord with the list that follows:

k = 0*	(D) → Q
k = 1	$(D_L) \rightarrow Y_L$
k = 2	$(D_L) \rightarrow Y_U$
k = 3	(D) → Y
k = 4*	(D) → A
k = 5	**$(D_L)' \rightarrow Y_L$
k = 6	$(D_L)' \rightarrow Y_U$
k = 7	(D)′ → Y

* A k=0 in instruction 14 causes (Q)′ → Q and k=4 in instruction 15, 32 and 33; (A)′ → A.

** The prime mark after a parenthesis (....)′ is defined as "take the complement of the contents of ....".

Interpretation of k in Replace Instructions

In replace instructions (f = 24-25, 34-37, 44-46, 54-57), k governs, first, the derivation of the operand as in read instructions, and secondly, the transfer of a word to address Y as in store instructions. (The values k = 0,4,7 are not used).

Exceptions

For instructions 13, 17, 62, 63, 66, 67, 73, 74, 75 and 76, the f, j*, k*, b and y designators are as follows:

f	j*	k*	b	y
29 24	23 20	19 18	17 15	14 0

Figure 4-2. Input-Output Instruction Word

j* = In these instructions j designates the input-output channels.

k* = In these instructions k is interpreted as "0" or not "0".

For instructions 40 and 44—Enter Logical Product and Replace Logical Product:

j = 2 Skip next instruction if the parity of the final contents of A is even.

j = 3 Skip next instruction if the parity of the final contents of A is odd.

In general Q is considered to contain the mask for which the parity is known. Then by choosing the appropriate mask for which the parity is known, the parity can be calculated on any selected group of bits in the accumulator as defined by the mask.

Illegal Instruction Code Fault

Instruction Codes 00 or 77 cause a jump to fixed address 00014 and initiation of the interrupt mode.

Special Address Allocation

The allocation of the special addresses for the control of buffers, external interrupts and internal interrupts are tabulated below:

00014	Fault, delta clock and millisecond time-out interrupt.
00017	Incremental—Incremental-Interrupt
00020-00035	External Interrupt
00040-00055	Input Internal Interrupt
00060-00075	Output Internal Interrupt
00100-00115	Input Buffer Control
00120-00135	Output Buffer Control

j DESIGNATORS FOR * COMMANDS

j	04	26 or 27	60
0	No Skip	No Skip	No Jump Deac. int. mode
1	Skip	Skip	Jump & Deac. int. mode
2	Skip If (Y) ≦ (Q)	Skip If (A) Pos	Jump If (Q) Pos
3	Skip If (Y) > (Q)	Skip If (A) Neg	Jump If (Q) Neg
4	Skip If (Q) ≧ (Y) And (Y) > (A)	Skip If (Q) = 0	Jump If (A) = 0
5	Skip If (Q) < (Y) Or (Y) ≦ (A)	Skip If (Q) ≠ 0	Jump If (A) ≠ 0
6	Skip If (Y) ≦ (A)	Skip If (Q) Neg	Jump If (A) Pos
7	Skip If (Y) > (A)	Skip If (Q) Pos	Jump If (A) Neg

TABLE 1

j DESIGNATORS FOR *COMMANDS

j	61	64	65	70
0	Jump	No Jump	Return Jump	Note A
1	Jump If JUMP 1 Is Set	Return Jump	Jump If JUMP 1 Is Set	Note B
2	Jump If JUMP 2 Is Set	Jump If (Q) Pos	Jump If JUMP 2 Is Set	Note C
3	Jump If JUMP 3 Is Set	Jump If (Q) Neg	Jump If JUMP 3 Is Set	Note D
4	Jump And Stop	Jump If (A) = 0	Jump, Stop	Note E
5	Jump; Stop If STOP 5 Is Set	Jump If (A) ≠ 0	Jump; Stop If STOP 5 Is Set	Note F
6	Jump; STOP If STOP 6 Is Set	Jump If (A) Pos	Jump; Stop If STOP 6 Is Set	Note G
7	Jump; Stop If STOP 7 Is Set	Jump If (A) Neg	Jump; Stop If STOP 7 Is Set	Note H

TABLE 2

y MODIFICATION OF REPEATED INSTRUCTION

NOTE A: IF j^{70} = 0 or 4, repeated instruction unmodified

NOTE B: IF j^{70} = 1 or 5, advance execution address of repeated instruction each execution except the first

NOTE C: IF j^{70} = 2 or 6, back execution address of repeated instruction each execution except the first

NOTE D: IF j^{70} = 3 or 7, add (B^b) of repeated instruction each execution

NOTES: E, F, G, H, are like A, B, C, D, respectively, with the added stipulation that if the repeated instruction is a replace instruction, (B^6) is added to the operand address for the store portion of the replace instruction.

j and k DESIGNATORS

	j Interpretation for f ≠ 04, 12, 13, 16, 17, 26, 27, 60-67 and 70-76	k Interpretation for ALL Instructions			
		Read	Store	Replace Read	Replace Store
0	No Skip	$(U)_L \rightarrow X_L$, $0 \rightarrow X_U$	$(X) \rightarrow Q$	Not Used	Not Used
1	Skip	$(Z)_L \rightarrow X_L$, $0 \rightarrow X_U$	$(X)_L \rightarrow Y_L$	$(Z)_L \rightarrow X_L$, $0 \rightarrow X_U$	$(X)_L \rightarrow Y_L$
2	Skip If (Q) Pos.	$(Z)_U \rightarrow X_L$, $0 \rightarrow X_U$	$(X)_L \rightarrow Y_U$	$(Z)_U \rightarrow X_L$, $0 \rightarrow X_U$	$(X)_L \rightarrow Y_U$
3	Skip If (Q) Neg.	$(Z) \rightarrow X$	$(X) \rightarrow (Y)$	$(Z) \rightarrow X$	$(X) \rightarrow Y$
4	Skip If (A) = 0	$(U)_L \rightarrow X_L$, $U_{14} \rightarrow X_U$	$(X) \rightarrow A$	Not Used	Not Used
5	Skip If (A) ≠ 0	$(Z)_L \rightarrow X_L$, $Z_{14} \rightarrow X_U$	$(X)'_L \rightarrow Y_L$	$(Z)_L \rightarrow X_L$, $Z_{14} \rightarrow X_U$	$(X)_L \rightarrow (Y)_L$
6	Skip If (A) Pos.	$(Z)_U \rightarrow X_L$, $Z_{29} \rightarrow X_U$	$(X)'_L \rightarrow Y_U$	$(Z)_U \rightarrow X_L$, $Z_{29} \rightarrow X_U$	$(X)_L \rightarrow Y_U$
7	Skip If (A) Neg.	$(A) \rightarrow X$	$(X)' \rightarrow Y$	Not Used	Not Used

TABLE 3

SUMMARY OF INSTRUCTIONS WITH TIMING

FUNCTION-CODE DESIGNATOR	INSTRUCTION	OPERATION	EXECUTION TIMES					
			NORMAL j = 0, 1			REPEAT j = 0		
			k = 0, 4	k = 7	k ≠ 0,4,7	k = 0,4	k = 7	k ≠ 0,4,7
01	Shift Q Right	Shift (Q) Right by (Y)	7.2 9.6	8.4 10.8	12			
02	Shift A Right	Shift (A) Right by (Y)	7.2 9.6	8.4 10.8	12			
03	Shift AQ Right	Shift (AQ) Right by (Y)	8.4 12	8.4 12	12 15.6			
04	Compare	Sense (j); $(A)_i = (A)_f$*	9.6	8.4	12	7.2	6.0	8.4
05	Shift Q Left	Shift (Q) Left by (Y)	7.2 9.6	7.2 9.6	12			
06	Shift A Left	Shift (A) Left by (Y)	7.2 9.6	7.2 9.6	12			
07	Shift AQ Left	Shift (AQ) Left by (Y)	8.4 12	8.4 12	12 15.6			
10	Enter Q	(Y)→Q	8.4	7.2	12	6.0	4.8	7.2
11	Enter Accumulator	(Y)→A	8.4	7.2	12	6.0	4.8	7.2
12	Enter B Register	$(Y) \rightarrow B^J$	6.0	7.2	12	3.6	4.8	7.2
13	Enter External Function	$(Y) \rightarrow C^J$	18		18			
14	Store Q	(Q)→Y	8.4		12	4.8		7.2
15	Store Accumulator	(A)→Y	8.4		12	4.8		7.2
16	Store B Register	$(B^J) \rightarrow Y$	8.4		12	4.8		7.2
17	Store Input Channel	$(C^J) \rightarrow Y$			12			
20	Add	((A) + (Y))→A	8.4	7.2	12	6.0	4.8	7.2
21	Subtract	((A) − (Y))→A	8.4	7.2	12	6.0	4.8	7.2
22	Multiply	(Q) (Y)→AQ		19.2		84		
23	Divide	(AQ/(Y)→Q, Remainder = $(A)_f$		84				
24	Add Replace	((A) + (Y))→Y & A			18			12
25	Subtract Replace	((A) − (Y))→Y & A			18			12
26	Q Add	((Q) + (Y))→Q; *$(A)_i = (A)_f$	9.6	8.4	12	7.2	6.0	8.4
27	Q Subtract	((Q) − (Y))→Q; *$(A)_i = (A)_f$	9.6	8.4	12	7.2	6.0	8.4
30	Load A Add Q	((Y) + (Q))→A	8.4	7.2	12	6.0	4.8	7.2
31	Load A Subtract Q	((Y) − (Q))→A	8.4	7.2	12	6.0	4.8	9.6
32	Add Q and Store	((A) + (Q))→Y & A	9.6		12	4.8		7.2
33	Subtract Q and Store	((A) − (Q))→Y & A	9.6		12	4.8		7.2
34	Replace Add Q	((Y) + (Q))→Y & A			18			12
35	Replace Subtract Q	((Y) − (Q))→Y & A			18			12
36	Replace Add One	((Y) + 1)→Y & A			18			12
37	Replace Subtract One	((Y) − 1)→Y & A			18			12
40	Enter Logical Product	L(Y) (Q)→A	8.4	7.2	12	6.0	4.8	7.2
41	Add Logical Product	(L(Y) (Q) + (A))→A	8.4	7.2	12	6.0	4.8	7.2
42	Subtract Logical Product	((A) − L(Y) (Q))→A	8.4	7.2	12	6.0	4.8	7.2
43	Masked Comparison	(A) − L(Y) (Q), Sense (j), (A) + L(Y) (Q); $(A)_i = (A)_f$	8.4	7.2	12	6.0	4.8	7.2
44	Replace Logical Product	L(Y) (Q)→Y & A			18			12
45	Replace Add Logical Product	(L(Y) (Q) + (A))→Y & A			18			12
46	Replace Subtract Logical Product	((A) − L(Y) (Q))→Y & A			18			12
47	Store Logical Product	L (A) (Q)→Y; $(A)_i = (A)_f$	9.6		12	4.8		7.2
50	Selective Set	Set $(A)_n$ for $(Y)_n = 1$	9.6	8.4	12	7.2	6.0	8.4
51	Selective Complement	Complement $(A)_n$ for $(Y)_n = 1$	9.6	8.4	12	7.2	6.0	8.4
52	Selective Clear	Clear $(A)_n$ for $(Y)_n = 1$	9.6	8.4	12	7.2	6.0	8.4
53	Substitute	$(Y)_n$ $(A)_n$ for $(Q)_n = 1$	9.6	8.4	12	7.2	6.0	8.4
54	Replace Selective Set	Set $(A)_n$ for $(Y)_n = 1$,→Y & A			18			12
55	Replace Selective Complement	Complement $(A)_n$ for (Y) = 1,→ Y & A			18			12
56	Replace Selective Clear	Clear $(A)_n$ for $(Y)_n = 1$,→Y & A			18			12
57	Replace Substitute	$(Y)_n \rightarrow (A)_n$ for $(Q)_n = 1$,→ Y & A			18			12
60	Arithmetic Jump	j = 0, No Jump; j ≠ 0, Jump*	6.0	7.2	12			
61	Manual Jump	Jump*	6.0	7.2	12			
62	Input Active Buffer Jump	(C^J) Active, Jump to Address (Y)	6.0		12			
63	Output Active Buffer Jump	(C^J) Active, Jump to Address (Y)	6.0		12			
64	Arithmetic Return Jump	J = 0, no Jump; J ≠ 0 Jump to Y + 1, (P)→Y*	9.6 14.4	8.4 13.2	12 18			
65	Manual Return Jump	Return Jump, Stop if STOP key up*	9.6 14.4	8.4 13.2	12 18			
66	Deactivate Input Buffer	Terminate Input Buffer designated by j.	6.0		12			
67	Deactivate Output Buffer	Terminate Output Buffer designated by j.	6.0		12			
70	Initiate Repeat	Execute NI (Y) Times*	6.0	7.2	12			
71	Index Skip	(B^J) = (Y): Skip NI, Clear B^J; (B^J) ≠ (Y): Adv. (B^J), Read NI	7.2	8.4	12			
72	Index Jump	(B^J) = 0: Read NI; (B^J) ≠ 0: (B^J) − 1, Jump to Address (Y)	6.0	7.2	12			
73	Initiate Input Buffer Without Monitor	Initiate Channel designated by j.	12		18			
74	Initiate Output Buffer Without Monitor	Initiate Channel designated by j.	12		18			
75	Initiate Input Buffer With Monitor	Initiate Channel designated by j.	12		18			
76	Initiate Output Buffer With Monitor	Initiate Channel designated by j.	12		18			

** See j Designators for * commands.*

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

PRINTED IN U.S.A. UT2403 REV. 1