UNIVAC SCIENTIFIC

general purpose computer system

PROGRAMMING

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

UNIVAC SCIENTIFIC GENERAL-PURPOSE COMPUTER SYSTEM

PROGRAMMING

PX 18

SEPTEMBER 1956

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

LIST OF ILLUSTRATIONS

INTRODUCTION

A computer is a device which is capable of (1) accepting data and instructions to perform operations on this data, (2) executing the specified operations, and (3) producing the results of the operations. More specifically, a general-purpose digital computer is capable of executing the basic arithmetic operations, performing internal data handling operations and logical operations, receiving data from a source external to the computer, and transmitting data to external media of representation. Any problem which can be solved by numerical techniques can be handled and solved by computer operations. The given problem must be analyzed and resolved into a collection of smaller problems, each of which can be solved by the application of the basic computer operations.

A general-purpose digital computer comprises electrical circuits, electronic and magnetic devices, and an associated power supply. Data which is to be manipulated internally is given a numerical representation. Such numbers are represented internally by a state or a condition of a component of the computer. Operations are performed upon the data by applying signals, effecting a change in its condition, to the device which holds the numerically coded data. The function assigned to such a signal is determined according to the change effected in the condition of the component. Data manipulations are performed by an ordered sequence of such impulses upon the components of the computer. The instigation of these impulses, and the regulation of the order in which they are initiated, are the functions of the <u>control</u> portions of a computer.

An operation (permissible to a particular computer) occurs when a portion of control detects a directive to begin the pre-determined sequence of steps necessary to effect the operation. Major directives (computer instructions) are given the computer in a prescribed coded numerical form by the operator of the computer. These directives, and the data which is to be manipulated in a computer operation, may be set aside within the computer, temporarily or for later use, in the <u>storage</u> section. When the information is needed, it is recalled from storage and placed in the appropriate section of the computer. If an arithmetic or logical operation is indicated, data is placed in the <u>arithmetic</u> section of the computer. Here the desired manipulations of the numbers are effected, and temporary storage for results is provided.

The <u>input output</u> portion of the computer consists of the components necessary to (1) provide insertion of coded data and directives into the computer and (2) present externally the results of computations carried out by the computer.

The <u>power</u> system of the computer provides regulated voltages to the four major sections of the computer discussed above.

Discussions of computer systems have led to the establishment of a computer "language". Certain basic terms in this language are discussed subsequently.

STAGE	- Electronic device which may be in unique states (conditions). The number of the states possible to the device determines the radix of the number system allowable to representation by the computer. It is possible for a stage in the Univac Scientific to have two unique stable states: a state which is assigned to represent "0", and a state which is assigned to represent "1". Thus, the radix of the number system used in the Univac Scientific is two; the number system is the binary system. A stage may receive as input electrical signals (pulses) which set it to its "1" state, set it to its "0" state, or reverse its present state. Signals derived from the stage indicate the state of the stage and are interpreted as a "1" output or a "0" output. This simple system of input and output to and from a bi-stable stage provides a means of adding, subtracting, and directly complementing binary numbers.
REGISTER	- A quantity of stages. The regulation number of stages determines the modulus of the number system allowable to representation by the computer system. The nature of a register allows its use as a storage device for information. Frequently, other storage devices for information are also referred to as "registers".
ADDRESS	- A coded number which designates specifically some particular computer register or other internal storage location or device. Information is referenced by its address. Portions of computer control are responsible for directing information to or from an addressed location.
BIT	- A binary digit, "0" or "1", represented in the computer by a state of a bi-stable medium of representation.
WORD	- Information coded for computer representation as a series of bits. The normal word length is considered to be 36 bits.
OPERAND	- A word representing coded data which is involved in computer operations or results from computer operations.
INSTRUCTION	- A 36 bit word which is a coded directive to the control section to initiate and oversee a prescribed sequence of steps necessary to effect a particular arithmetic, logical, or input output operation. Portions of a computer instruction designate the operands which are involved in, and necessary for, the execution of the particular operation.
OPERATION CODE	- The coded portion of the instruction which "describes" to computer control which particular operation is to be executed.
INSTRUCTION-FUNCTION	An explanation of what the execution of each instruction accomplishes, with the locations specified of (1) any operands used during this execution and (2) any pertinent results derived from this execution.
PROGRAM	- A sequence of coded computer instructions and necessary operands for the solution of a problem.

INPUT OUTPUT	- systems providing the means of communication between the computer and the operator. Input and output operations involve units of external equipment control, certain of the computer registers, and portions of the computer control section.
STORAGE	- consists of devices in which information is set aside for immediate or future use. Each storage location in the Univac Scientific has a unique.address. Each location in the storage section of the computer is an arrangement of 36 bistable elements; hence, each location is capable of storing 36 binary digits.
ARITHMETIC	- a section where arithmetical and logical operations are performed and operands and results temporarily stored.
CONTROL	- consists of components which direct the operations of the computer.

DESCRIPTION OF THE COMPUTER

1. GENERAL

The basic concept of the solution of a problem on a computer is presented in the following paragraphs. First, the computer instructions and their functions must be studied in order to gain a thorough understanding of the capabilities and restrictions of each instruction. When this has been achieved, the problem to be solved is reduced to a sequence of simplified steps, the arithmetic and logical operations of which can be solved by application of the instructions. The Univac Scientific has in its repertoire 41 instructions, each of which is represented by 36 bits, $i_{35}...i_0$. The left-most six bits of an instruction, $i_{35}...i_{30}$, represent its operation code. The remaining 30 bits are grouped as $i_{29}...i_{15}$ and $i_{14}...i_0$. These bits are designated as the u address portion and v address portion of the instruction, respectively. These are the portions of the instruction which represent the operands (by referencing their location in storage, for the most part) with which the operation is concerned. A program is prepared by arranging the instructions in the order in which their operations are desired. The instructions are not written in binary but are coded in octal, each octal digit representing three binary digits. Thus, twelve octal digits represent an instruction; two digits represent the operation code; five digits, the u address; and five digits, the v address. For example, the instruction termed "Transmit Positive", whose function is in general "Replace the information at a certain specified location v with the information from another specified location u," is coded abstractly as 11 uv. With specific locations assigned to u and v, for instance, those with the numerical assignments of 01000 and 00100, the instruction is written in octal as

11 01000 00100.

This represents the binary notation of

001001000001000000000000000001000000.

This instruction, as coded, directs the computer to "replace the content of location 00100 with the content of location 01000".

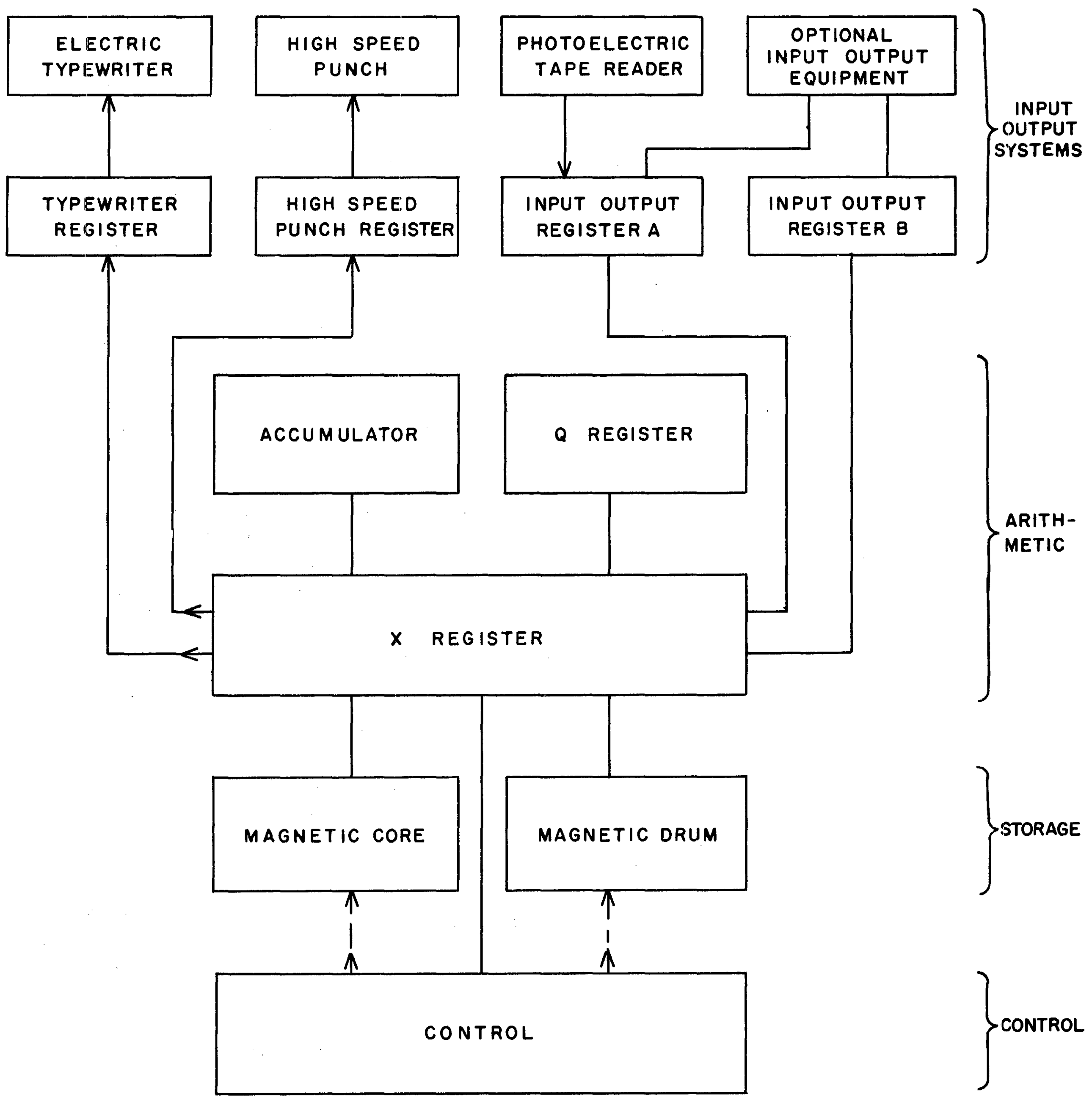
When the program has been prepared, the coded list of instructions, operands, and any other data comprising it, are prepared for "loading", i.e., entrance into the computer. An input procedure results in the program being stored in the computer at a series of consecutive locations, the first of which must be specified during the input procedure. Each instruction and operand which was coded as a 12 octal digit number is stored at an individual location as a 36 binary digit number. Operands referenced by instructions must be stored at the address specified by the instruction. For instance, the information to be transferred from one location to another by the Transmit Positive instruction, coded as 11 01000 00100, must actually be in storage at the location addressed as 01000. During its execution, a program is usually in storage in a "rapid access" (high speed) type of storage, but it may or may not be placed there initially during the loading process. Rapid access storage allows the fastest possible execution of a program since it provides the quickest acquisition of a word from storage when it is needed.

The order in which the instructions are taken from storage and executed is regulated by the computer control section. Instructions are chosen for execution according to the content of a 15-bit register in the control section. This register holds the storage address of the instruction to be executed. To start the execution of a program, the address of the instruction to be first executed is placed in this register. A controlled series of electrical impulses then (1) "review" the state of the components of the register, determining its content, (2) direct the process of referencing this location, (3) detect the information stored at this address, and (4) transfer it to a 36-bit register in control reserved for holding an instruction during its execution. An instruction remains in this register until it is replaced by the instruction to be executed next. The address of this instruction is again taken from the 15-bit address register. This 15-bit control register has a "counting" facility. Normally, the number represented in the register is advanced by one after each reference to storage is made. This procedure results in the automatic acquisition of instructions for execution from consecutive storage locations.

The presence of the 36 bits of an instruction in the control register reserved for it, authorizes and enables control to direct the execution of the instruction. The particular state of the six left-most stages of the register (those holding the operation code of the instruction) allows a certain sequence of pulses to be released to a portion of the control circuitry. Each pulse in this sequence initiates a series of pre-determined operations. The particular series inaugurated depends upon the combination of bits comprising the operation code. When, during the series of operations, the information at the u or v addressed portion of the instruction is needed, a review occurs of the states of the stages 29 through 15, or 14 through 0, of the register holding the instruction. The process then is to determine this address, reference it, and transfer the data from this location to the register where the data is needed.

All computer operations depend primarily on sequences of pulses originating in the control section. The pulses are issued regularly from a "clock source" at the rate of one every two microseconds. Their release, and the time of their release, to various portions of the control section and to other sections of the computer depends upon the current status of computer operation. This conditional initiation of any sort of a computer operation effects an orderly progression of the steps involved in the execution of an instruction and a series of instructions.

The basic functional make-up of the Univac Scientific General-Purpose Computer System is illustrated in Figure 1 in which the major sections, input output, arithmetic, storage, and control are delineated. A brief discussion of some of the components of the storage, arithmetic, and control sections follows in the subsequent paragraphs. A discussion of the input output systems is found later in the text.



Solid lines connecting blocks indicate the routing of binary information. Dashed lines carry binary information used for storage reference purposes. Information may be routed in both directions unless arrows on a line indicate a one-way transfer.

Lines which carry control signals between the above blocks are not shown on this diagram.

Figure 1. Simplified Block Diagram of the Univac Scientific General-Purpose Computer System

2. PRINCIPAL REGISTERS

A large quantity of registers, providing static storage of information, is necessary in the computer to facilitate the enumerable computer operations. The number of stages comprising a register depends upon the function that is served by the register. For instance, a register intended to hold a computer instruction consists of 36 stages; a register intended to hold a 15 binary digit address consists of 15 stages. Operations which use in some fashion the content of a register affect all the stages of the register simultaneously. This is described as the "parallel mode" of operation, as opposed to the "serial mode". In the serial mode of operation any sampling, setting, or reversing the states of the stages of a register, proceeds cyclicly one stage at a time.

In addition to the various control registers which are involved in the execution of an instruction, there are three other principal registers which are involved quite frequently. These registers are listed below.

a. X REGISTER. - The X Register is used during the execution of any instruction whose u or v address references an addressable location. The information addressed by u or v is transmitted from its location to the X Register, and from the X Register to the desired position in the computer. Thus the X Register serves as a transmission register for most internal routing of information. In addition the X Register functions as a component of the arithmetic section of the computer. For the corresponding arithmetic register, the X Register holds the addend, subtrahend, multiplicand, and divisor.

   In general, the X Register, abbreviated as X, is a 36-stage register capable of temporary storage of 36 bits of information.

b. Q REGISTER. - The Q Register functions as a component of the arithmetic section and also serves as an addressable storage device. The Q Register, designated as Q, comprises 36 stages, affording temporary storage to 36 bits of information.

   For the corresponding arithmetic operations, the Q Register holds the multiplier, quotient, and logical multiplier. The Q Register derives its nomenclature from the use of this register for the assembly of the quotient during a divide operation.

   The Q Register has "shift left" with "end-around shift" facilities. When a "shift left once" operation is completed, each stage of the register is in the state which was reflected by the stage immediately to the right of it before the operation began. The right-most stage of the register is set to the state of the left-most stage. In other words, the binary digits held in a register are displaced to the left, as many places as is specified by the shift operation, with the left-most bit being shifted in a circular fashion to the right end of the register.

c. ACCUMULATOR. - The Accumulator functions as a component of the arithmetic section and also serves as an addressable storage device. The Accumulator, referred to by the letter A, comprises 72 stages, affording temporary storage to 72 bits of information. Because the Accumulator is capable of holding twice the number of bits of an ordinary storage register, it is often referred to as a "double-length" register. The left-most 36 stages of the Accumulator are referred to collectively as A_L; the right-most 36, as A_R.

For the corresponding arithmetic operations, the Accumulator holds the sum, difference, product, dividend, and (at the end of a divide operation) remainder. The Accumulator derives its nomenclature from the use of this register for the accumulation of sums. The double length feature of this register allows the formation of sums of more than 36 bits. It also allows the formation of the full product of any two 36 bit numbers, regardless of their numerical value; and conversely, it allows the formation of a quotient whose numerical value is the greatest possible (in absolute value) to represent in a 36-bit register.

The Accumulator also has the "circular shift left" property described for the Q Register. The 72 bits held in the register are displaced to the left, as many places as is specified by the shift operation, with the left-most bit being shifted in a circular fashion to the right end of the register.

The contents of a register, i.e., the bits represented by the states of its stages, is designated by enclosing the symbol for the register in parenthesis. For example, the 36-bit content of the X Register is denoted as (X); the content of the 36 right-hand stages of the Accumulator is denoted as (A_R).

3. STORAGE DEVICES

The information which is held in storage consists of instructions which are to be executed and the operands needed by these instructions for their execution. The location of each instruction or operand is referred to as its "address".

a. ADDRESSED STORAGE LOCATIONS. - There are four classes of storage locations which are individually addressed: Magnetic Drum Storage, MD; Magnetic Core Storage System, MCS; the Accumulator, A; and the Q Register, Q. These classes have the following addresses assigned to them:

Storage Class	Octal Equivalents of Addresses	Storage Space: Number of Words
MCS-0	00000-07777	4096
MCS-1	10000-17777	4096 optional
MCS-2	20000-27777	4096 optional
Illegal Addresses	30000-30777	
Q	31000-31777	1
A	32000-37777	1 double length
MD (Group 4)	40000-47777	4096
MD (Group 5)	50000-57777	4096
MD (Group 6)	60000-67777	4096
MD (Group 7)	70000-77777	4096

The banks of Magnetic Core Storage, MCS-1 and MCS-2, are optional to the computer system. If this storage is not provided, the addresses assigned to MCS-1 and/or MCS-2 are illegal addresses.

Additional storage, in which the information is not individually addressed, is provided by up to ten Uniservo magnetic tape units.

Information is acquired from storage in MCS, MD, A, or Q by first determining the storage class of the address of the instruction or operand desired. Following this, the address is transmitted to the locating control of the proper storage class where the specific storage location is found (if the storage class is MC or MD). Then the information at this location is transmitted to the X Register. Operations of this kind are referred to as "reading" operations. "Writing" operations, or the transfer of information to a storage location, are accomplished in a similar manner with the information in the X Register being placed at a storage address as located by the control circuitry.

b. MAGNETIC DRUM STORAGE SYSTEM (MD). - The Magnetic Drum Storage System provides medium-access binary storage. Digital information is stored in the form of magnetized areas on the surface of a continuously rotating cylinder called a magnetic drum. The medium of storage is a magnetized bipole having either of two polarity orientations in the lateral (or peripheral) direction. For all practical purposes, information recorded on the drum is stored permanently. It may, however, be removed by special erase techniques, or it may be replaced by simply writing new information over it. Reading from the drum does not in any way alter the contents of the location read.

Each individual storage location is identified by specifying its angular and axial coordinates on the drum surface. The 36 bits of a word are stored at 36 individual axial positions on the drum. A drum group has angular storage space available in normal drum operation for 4096 36-bit words. A total of four drum groups results in an MD storage capacity of 16,384 words. When a word or a portion of a word is to be transmitted to or from the magnetic drum, all the bits to be transmitted are handled simultaneously, i.e., in parallel. Information may be recorded or read in any given area only once during each drum revolution, resulting in a maximum access time of 34 milliseconds.

The first octal digit of an MD address (4, 5, 6, or 7) specifies the drum group or axial location of a word. The remaining four octal digits specify the angular address (0 through 7777 octal) of the word in the designated group. The angular locations during a revolution of the drum are counted and recorded by the Angular Index Counter. An MD reference made during a computer operation is translated according to a chosen "interlace", and held in an address interlace chassis. When coincidence is detected between these two MD addresses, the reading or writing operation occurs. According to the pre-selected interlace, the reading or writing occurs at a regulated interval from the actual drum location originally referenced. Interlaces of 4, 8, 16, 32, or 64 are available. The address held in the address interlace chassis is the modular product (in binary) of the angular portion of the MD address referenced, and the power of two specified by the interlace. If an interlace of four is chosen, coincidence occurs between consecutive MD references and the address of every fourth MD location, etc. For example, if an interlace of eight is chosen, an MD reference listed below in the left-hand column results in the selection of the MD location listed below in the right-hand column.

Original Reference	As held in the Address Interlace Chassis
40000	40000
40001	40010
40002	40020
.	.
.	.
.	.
40777	47770
41000	40001
41001	40011
.	.
.	.
.	.
41777	47771
.	.
.	.
.	.
47000	40007
47001	40017
.	.
.	.
.	.
47777	47777

The variable interlace system allows the selection of the minimum computer time for consecutive MD read and write operations. The time required for one drum revolution is 34 milliseconds. Thus, the time which elapses between the positioning of physically adjacent drum locations for read or write operations is approximately eight microseconds. An interlace of "1" would mean that references to two consecutive MD addresses would have to be made in less than eight microseconds for the drum to be properly positioned for the second read or write operation before a complete drum revolution has ensued. If the references are made in less than 32 microseconds, a four interlace effects the minimum time possible for the MD read or write operations. The interlace which is in effect is indicated by an illuminated light on the upper right section of the Supervisory Control Panel of the computer.

The preceding paragraphs were written with the normally addressable MD storage locations in mind. Each drum group has, in addition to the previously mentioned 4096 locations for storage, a "reserve space" of 160 locations which are not normally addressable. Communication is established with these locations, and broken with the rest of the drum locations, by setting the NORMAL/ABNORMAL DRUM switch on the lower right section of the Supervisory Control Panel to ABNORMAL. This will allow the detection of coincidence with reserve space locations zero through 0237 (octal). When the normally addressed portion of the drum is in position for reading or writing, the Angular Index Counter counts from zero to 7777 (octal) but no coincidence tests are made. When the reserve space on the drum is in position for reading or writing, the Angular Index Counter counts from zero to 0237 (octal) and checks for coincidence with addresses in the address interlace chassis. The MD references made for Abnormal drum reading or writing must be properly coded so that their form in the interlace chassis will be octal -0000 through -0237. (The first octal digit may be 4, 5, 6, or 7 depending upon the drum group to be addressed.)

c. MAGNETIC CORE STORAGE SYSTEM (MCS). - Each bank of the Magnetic Core Storage System provides rapid-access storage for 4096 36-bit words. Each core is a bistable element capable of storing a "1" or a "0", dependent upon the direction of magnetization of the core. The cores are arranged in a 64 x 64 matrix with 36 such matrices. The 36 digits of a given word are represented by the state of 36 corresponding cores, one in each of the 36 matrices. Reading and writing operations of a word, or portion of a word, are performed in a parallel mode with a simultaneous transmission of bits. Certain sequences of pulses on wires through the cores, producing magnetizing forces of a certain polarity, are used to perform the reading and writing operations. Reading from MC does not in any way alter the contents of the location read.

Magnetic Core Storage is non-volatile; comparable to non-volatile storage in the Magnetic Drum Storage System.

d. A AND Q AS STORAGE MEDIA. - The Accumulator and Q Register are available as temporary storage registers since they may be addressed. The Q Register is normally addressed as octal 31000 although any of the addresses 31000-31777 are permissible. Similarly, the Accumulator is normally addressed as 32000 with the addresses 32000-37777 being permissible.

4. CONTROL COMPONENTS.

Each of the function groupings of the computer, input and output, storage, and arithmetic, has individual control systems which direct the operations of the section under their influence. These control systems are in turn directed in their operations by the main computer control. This overall influence exerted by computer control is necessary for time-wise reasons: an established sequence of internal actions is essential for the processing of any coded information. The computer control initiates and superintends these patterns of actions during their performance.

The main control section receives the instructions which the computer is to carry out; it interprets them, and directs their execution with the operands specified. The computer must be manually started, but can be either automatically or manually stopped. (In addition to being automatically controlled by a program of instructions, the computer can be manually controlled from the Supervisory Control Panel which contains all the necessary controls and indicators for manually operating the equipment.)

The principal components of the control section are as follows.

a. PROGRAM ADDRESS COUNTER. - The Program Address Counter, PAK, is a 15-stage additive register. During computation PAK generates the consecutive addresses of the programmed instructions to be executed. The address in PAK is normally referred to each time an instruction word is to be obtained from the computer memory. The starting address for a computation may be manually inserted into PAK before the START (operation) button is pushed. If this is done, computation will begin by picking up the instruction stored at that address. If PAK is not manually pre-set, it will automatically be set to MD address 40000. During the normal termination of an instruction, the next instruction to be executed (the address of which is held at that time in PAK) is extracted from storage; and the content of PAK is advanced by one. Thus, during the termination of the instruction at address y, the instruction at y + 1 (the address held in PAK) is extracted from storage, and PAK is advanced to y + 2. If the instruction at address y indicated that the sequential acquisition of instructions be disrupted by a jump to an instruction not stored at a consecutive address, this instruction's address is inserted into PAK previous to the termination operations.

The program interrupt feature of the Univac Scientific, discussed in a later paragraph, interrupts the normal process of acquiring the address of the next instruction from PAK.

The generation of consecutive binary numbers in PAK is restricted by the following conditions in its physical structure. There is no communication between the stages PAK_{12} and PAK_{11} unless the stage PAK_{14} contains a value of one. Thus, PAK_{12} will not be affected by the advance of PAK after the contents of PAK_{11} ... PAK_0 reach the value of $2^{12}-1$ (7777 octal). The next advance of PAK, after such a value is reached, results in the contents of the stages PAK_{11} ... PAK_0 being changed to zeroes. If PAK_{14} does contain a one, the contents of PAK may be increased until the contents of PAK_{13} ... PAK_0 reach the value of $2^{14}-1$. Then, since there is no communication between the stages PAK_{13} ... PAK_0 being changed to zeros with the value of one being left undisturbed in PAK_{14}. This "closed loop" system effects the generation of successive MCS addresses in PAK as follows: the addresses of each bank of MC can be advanced to (octal) -7777, with the next advance of PAK resulting in its contents becoming (octal) -0000. If any of the Magnetic Drum addresses, regardless of the group, are represented in PAK, the addresses can be generated consecutively to 77777 with the next advance of PAK resulting in its contents becoming 40000.

b. PROGRAM CONTROL REGISTERS. - The Program Control Registers, PCR, receive each instruction and temporarily store it during its execution. The registers consist of the Main Control Register, MCR, the U Address Counter, UAK, and the V Address Counter, VAK. Each instruction sent to PCR consists of a 6-bit operation code which is stored in MCR, a 15-bit u address portion which is stored in UAK, and a 15-bit v address portion which is stored in VAK. Each instruction is obtained from some 36-bit storage location as specified by the Program Address Counter, PAK. The physical structures of UAK and VAK are similar to that of PAK. An additional restriction on the generation of consecutive binary numbers in UAK and VAK is as follows. If an A or Q address is in UAK or VAK, it is not possible to advance the content of the stages zero through eight of UAK or VAK beyond 2^9-1 (octal 777). Thus, A or Q addresses are generated from octal 32000 to 32777 to 32000 and from octal 31000 to 31777 to 31000, respectively. The generation of consecutive MC and MD addresses is the same as in PAK.

c. MASTER CLOCK. - All the activities which take place within the computer, except for certain ones in the output sections, are synchronized by a central timing system, called the Master Clock. During NORMAL computer operation, the clock generates 500 kc clock pulses based on timing pulses from the Magnetic Drum Storage System, and after exerting certain controlling influences over them, supplies them to circuits throughout the computer. During TEST operations, a 500 kc oscillator may be used instead of the drum as the basic source of timing pulses.

d. MAIN PULSE DISTRIBUTOR. - The Main Pulse Distributor, MPD, receives clock pulses and distributes them in sequences of from four to eight pulses to the Command Timing Circuits. The distributor supplies each of the pulses sequentially on its eight output lines. In an eight pulse cycle, all of the output lines are used, and the pulses are designated, in the order of their generation, MP0 through MP7. The selection of a particular cycle is made on the basis of the operation code held in the Main Control Register, MCR. Each code selects the sequence which will permit the performance of the generation in the least possible time.

e. MAIN CONTROL TRANSLATOR. - The principal translator of the Main Control Translator, MCT, receives a 6-bit operation code from the Main Control Register and produces accordingly a single operation code "enable". In the Command Timing Circuits, the enable from MCT is used in the selection of the sequence of commands which are needed to execute the instruction currently in the Main Control Register. In the Main Pulse Distributor, the MCT enable is used in the selection of the sequence of main pulses required for the operation.

f. COMMAND TIMING CIRCUITS. - The Command Timing Circuits, CTC, produce a discrete sequence of commands which execute the specified operation. The commands initiated are chosen by combining the operation code enable from the Main Control Translator and the pulse cycle received from the Main Pulse Distributor. A pulse cycle consists of two or more of the pulses MP0 through MP5, and MP6 and MP7. It initiates the commands which execute the operation on pulses MP0 through MP5; reads the instruction to be executed next from storage into the X Register on MP6; then transfers the instruction from X to PCR on MP7.

g. PROGRAM INTERRUPT CONTROL. - An interrupt selection interferes with the execution of the normal termination commands occurring on MP6. The normal termination commands take the address of the next instruction to be executed from PAK and then advance PAK. With the interrupt in effect, the address 00002 in Rapid Access Storage, F_3, is chosen as the address of the next instruction to be executed. This instruction is read from storage to the X Register. On MP7 the normal transfer of the content of X to PCR is made. This puts the instruction at F_3 in position for execution and leaves the address in PAK undisturbed. Thus, for example, if the interrupt becomes effective during the execution of an instruction at address y, the address y + 1 in PAK (or u or v if the instruction being executed calls for a jump) is undisturbed during MP6, and (F_3) is taken as the next instruction. By appropriate programming, the content of PAK may be inserted in a temporary storage location and later referred to in such a way as to return operation to the instruction stored at the address in PAK.

The selection of an interrupt is effective only on an MP6 generated during the normal termination commands. This means that the selection of an interrupt during the repeated execution of an instruction is not effective until the repeating is brought to a conclusion, either by the execution of the instruction n times or by the occurrence of a jump. (This is discussed in more detail under the Repeat instruction, Sequential Presentation of Instructions section.) The interrupt selection may be made manually from the Supervisory Control Panel or as a function of input output operations. Selecting an interrupt option during input output operations is discussed later in the Input Output section. Briefly, an interrupt selection may be made for input output operations by appropriate programming or a manual setting on the piece of external equipment involved.

5. REPRESENTATION OF NUMERICAL VALUES

The bi-stable characteristic of the elements of the computer dictates the use of binary number notation in the representation of information. However, the computer cannot determine whether an array of bits is an instruction, data with numerical value, or data coded in some arbitrary fashion. If an array of digits is confronted in a register normally reserved for holding an instruction, the computer will try to treat it as an instruction; if an array of bits is confronted in an arithmetic operation, the computer will deal with it as having numerical value.

The computer treatment of an array of bits in arithmetic operations assumes the assignment of a numerical value to the bits as follows: the left-most bit of an array determines the sign of the number; a "1" designates a negative value; a "0" designates a positive value. The remaining bits of the array determine the absolute value of the number.

One's complement notation is used for expressing the negative of a quantity. The one's complement of a binary digit is the digit subtracted from the value of one. The one's complement of a digit represented by the state of a bi-stable element is formed by merely reversing the state of the element. In a number system which includes all the possible combinations of "0's" and "1's" from 000 ... 000 to 111 ... 111, positive quantities are represented by the combinations in which the left-most bit is zero, 000 ... 000 to 011 ... 111. The negatives of these quantities are represented by their one's complement, the combinations in which the left-most bit is one, 111 ... 111 to 100 ... 000. This left-most bit is termed the sign-bit of the number.

The bits representing a number are held in an arrangement of bi-stable elements, such as the stages of a register. The designation of the left-most element of k elements is given the subscript k-1; the designation of the adjacent element is given the subscript k-2; and continuing to the right, the designation of the right-most element is given the subscript 0. For example, the 36 stages of the Q Register are designated as Q_{35}, Q_{34}, ..., Q_1, Q_0. In general, the stage S_{k-1} of a k stage register holds the sign bit of a number, and the absolute value of the number is determined by the contents of stages $S_{k-2},\ldots,S_0$. If the sign bit is "0", the bits in stages $S_{k-2},\ldots,S_0$ are the coefficients a_i of a binary number $a_{k-2}2^{k-2} + a_{k-3}2^{k-3} + \ldots a_1 2^1 + a_0 2^0$. If the sign bit is "1", the one's complement of the bits in stages $S_{k-2},\ldots,S_0$ are the coefficients of the absolute value of the number. (The term "most significant bit" is given to the first digit from the left which differs from the digits to its left.) The values possible to represent in k bi-stable elements are in the range with the limits of $\pm(2^{k-1}-1)$, inclusive. In a

36 stage register, the limits are $\pm(2^{35}-1)$; in the 72-stage Accumulator, the limits are $\pm(2^{71}-1)$. The modulus of a number system represented by k stages is 2^k. However, if "plus zero" and "minus zero" are treated as a unique quantity, as is the case in the Univac Scientific, the modulus is 2^k-1.

The assumption of the binary point to the right of the right-most bit means that all numbers are considered as integers. This does not mean, however, that numerical operations are restricted to integers only or to integers in these ranges. A binary number s may be expressed as $s = s_1 2^{s_2}$. If s_1 and s_2 may be expressed as integers with values in the range appropriate for their placement in the computer, s_1 and s_2 may represent in the computer the number s. Numerical operations involving s and t $(t = t_1 2^{t_2})$ are performed machine-wise by the proper arithmetical procedures involving s_1 and t_1, and s_2 and t_2. Fractions may be represented machine-wise by integral values of s_1 and s_2; s_2 being negative. If s is scaled to its maximum representation by 36 bits such that $2^{35} > |s_1| \geq 2^{34}$, the number s is said to be "normalized".

To summarize the preceding discussion, numerical quantities, represented in the computer by k bi-stable elements, are integers of a binary number system. A negative number N of this system is represented in one's complement notation as $2^k-1 - |N|$. The range of integers I possible to represent in a 36 stage register is

$$1-2^{35} \leq I \leq 2^{35} -1;$$

in the 72-stage Accumulator,

$$1-2^{71} \leq I \leq 2^{71} -1.$$

6. ARITHMETIC OPERATIONS.

a. GENERAL. - The modulus of the one's complement binary system, as involved in arithmetic operations in the computer, is 2^k-1 where k is the number of stages in the registers involved in the operations. If the registers involved in the operations consist of 36 stages, the modulus of the number system is $2^{36}-1$. If the Accumulator is involved in the operations, the modulus of the system is $2^{72}-1$. This modulus of 2^k-1 (instead of 2^k) results from the generation during arithmetic operations of a unique representation of zero, i.e., each bi-stable element in the "0" state. The generation of a negative zero representation, i.e., each bi-stable element in the "1" state, is not possible.

When an instruction necessitates the transmittal to the Accumulator of a 36-bit integer, the conditions are established during the operation that change the modulus of the integer from $2^{36}-1$ to $2^{72}-1$. This is effected by assuming the existence of 36 bits to the left of the sign-bit. This "72-bit" integer is then transmitted to the Accumulator. The final modular contents of A reflects the value of the "72-bit" integer according to the nature of the transmitting operation. If it is desired that the <u>one's complement signed value</u> of the 36-bit integer be retained, the operation assumes that each of the simulated 36 left-hand bits has the value of the sign bit. Such an extension of a 36-bit number is designated as a double length extension, D(L), where L is the location address of the 36-bit number and (L) is the content of that address. If it is desired that the <u>value of the machine expression</u> of the 36-bit integer be left undisturbed by the transmitting operation, the value of the sign-bit of (L) is disregarded, and the assumption is made that 36 zeroes exist to the left of the sign-bit of the number. This "72-bit" number is designated as S(L), a split double-length extension.

When one of the above transmissions to the Accumulator is required, the 36-bit integer is first placed in the X register (by the operations of whichever instruction is being executed). (The only means of information transfer to or from A is via X.) Then, according to the instruction being executed, a machine sequence is performed which adds or subtracts one of the double length extensions to or from the content of the Accumulator.

Most of the arithmetic operations in the computer are accomplished by combinations of such computer commands as listed below. The commands themselves, which are internal computer directives, are instigated as the result of computer control interpreting a particular instruction operation code. Some of the more commonly used commands are as follows:

Clear X
Complement X

Clear Q
Shift Q

Clear A
Clear A_R
Clear A_L
Shift A

Commands which direct transmissions between X, Q, and A

A series of subcommands which in different combinations accomplish the following:

Add D(X) to A
Subtract D(X) from A
Split Add X to A, i.e., Add S(X) to A
Split Subtract X from A, i.e., Subtract S(X) from A

(1) CLEAR X, Q, OR A. - Commands which direct the clearance of any register result in each stage of the register being set to its zero state.

(2) COMPLEMENT X. - The command which directs the complementation of the content of the X Register reverses the state of each stage of the register. The complement of the content of X is denoted as (X')

(3) SHIFT Q OR A. - These commands effect the left shift of the bits in the register the number of places prescribed by the instruction and established internally in a shift counter. A shift left of k places is equivalent to a right shift of 36-k or 72-k places. A left shift effects a modular multiplication by a power of two. A "right shift" is equivalent to a modular division by a power of two. The word modular is emphasized because of the circular shifting feature which effects the shift of the bit represented in the left-most stage to the right-most stage.

The operations of addition and subtraction are treated subsequently.

b. ADDITION AND SUBTRACTION. - The fundamental arithmetic operation of the computer is subtraction. The Accumulator is termed a subtractive Accumulator because all additions and subtractions are performed by a subtractive process. The initial content of A, $(A)_i$, is the minuend, and the final content of A, $(A)_f$, is the remainder. This subtractive process is used as the basis of all arithmetic operations involving addition and subtraction. This prevents, as the result of an arithmetic operation, the representation of zero by a one in each stage of the Accumulator.

The process of subtraction necessitates an ability to borrow from a left-hand digit or digits. Machine-wise this is made possible by the parallel construction of the stages of the Accumulator. The end-around borrow as discussed in Appendix A of this volume, is a feature of subtraction in the Accumulator: a borrow propagated past the stage A_{71} is applied to the stage A_0.

The number to be added to, or subtracted from, the content of the Accumulator is placed in the X Register. Then, according to the operation desired, a sequence of actions occurs which leaves in the Accumulator the desired answer. If a subtraction sequence is executed, the remainder is reflected in the final content of the Accumulator as the initial content of the Accumulator minus one of the double-length extensions of the content of X, i.e.,

$$(A)_f = (A)_i - D(X)$$

or

$$(A)_f = (A)_i - S(X).$$

If an addition sequence is executed, the sum is reflected in the final content of the Accumulator as the initial content of the accumulator minus the complement of one of the double-length extensions of the content of X, i.e.,

$$(A)_f = (A)_i - D(X)'$$

or

$$(A)_f = (A)_i - S(X)'.$$

Any references in this text to the addition of a number to the Accumulator should thus be interpreted as the process of subtracting the complement of the number from the Accumulator.

Actually, the machine subtraction sequences use procedures of an addition sequence after an appropriate complementation of (X).

The general procedures of the four addition and subtraction sequences are listed below with examples of the operations given to the left. The examples use an X Register of four stages and an Accumulator of eight stages. These sequences use the same internal subcommands in different combinations according to the operation desired and the content of the X Register.

ADD X TO A

	$(A)_i$	=	0000	0110	(=+6)
	$(X)_i$	=		1100	(=-3)
Assume the existence of D(X)	$D(X)_i$	=	1111	1100	
Subtract complement of D(X) from A	$D(X)_i'$	=	0000	0011	
			0000	0110	
	minus		0000	0011	
	$(A)_f$	=	0000	0011	(=+3)

SPLIT ADD X TO A

	$(A)_i$	=	0000	0110	(=+6)
	$(X)_i$	=		1100	
Assume the existence of S(X)	$S(X)_i$	=	0000	1100	(=+12)
Subtract complement of S(X) from A	$S(X)_i'$	=	1111	0011	
			0000	0110	
	minus		1111	0011	
			0001	0011	
	borrow			1	
	$(A)_f$	=	0001	0010	(=+18)

SUBTRACT X FROM A

	$(A)_i$	=	0000	0110	(=+6)
	$(X)_i$	=		1100	(=-3)
Complement (X)	$(X')_i$	=		0011	
Assume the existence of D(X)	$D(X')_i$	=	0000	0011	
Subtract complement of D(X) from A	$D(X')_i'$	=	1111	1100	
			0000	0110	
	minus		1111	1100	
			0000	1010	
	borrow			1	
	$(A)_f$	=	0000	1001	(=+9)

SPLIT SUBTRACT X FROM A

	$(A)_i$	=	0000	0110	(=+6)
	$(X)_i$	=		1100	
Assume the existence of S(X)	$S(X)_i$	=	0000	1100	(=+12)
Complement S(X)	$S(X)_i'$	=	1111	0011	
Subtract complement of S(X) from A	$[S(X)_i']'$	=	0000	1100	
			0000	0110	
	minus		0000	1100	
			1111	1010	
	borrow			1	
	$(A)_f$	=	1111	1001	(=-6)

c. MULTIPLY SEQUENCE. - Multiplication performed machine-wise uses the shifting facilities of the Accumulator and the Q Register, and the Add X to A sequence. The execution of an instruction which orders a machine multiplication places the multiplier in the Q Register, Q, places the multiplicand in the X Register, X, and forms the product in the Accumulator, A. The product is formed by adding the multiplicand, or D(X) machine-wise, the appropriate number of times, as determined by the bits of the multiplier, (Q), into the Accumulator. The procedure is as follows.

Repeat 36 times:

1. Shift (A) left one place.
2. If the current (Q_{35}) is 1, add D(X) to (A).
3. Shift (Q) left one place.

The result would be the formation of the sum

$$\left\{\left\{\left(Q_{35} \cdot D(X)\right) 2 + Q_{34} \cdot D(X)\right\} 2 + \ldots + Q_1 \cdot D(X)\right\} 2 + Q_0 \cdot D(X).$$

Thus, the multiplication of a number in X by a number in Q results in a sum in the Accumulator of

$$(Q_{35})\ (X)\ 2^{35} + (Q_{34})\ (X)\ 2^{34} + \ldots\ (Q_0)\ (X)\ 2^0.$$

An example of the multiplication process follows, using four bit Q and X registers and an eight bit Accumulator. The machine-wise formation of a sum by complementation and subtraction is not shown.

(X) = 0011, multiplicand of decimal 3

(Q) = 0101, multiplier of decimal 5

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Stages of A
0	0	0	0	0	0	0	0	initial content of A
0	0	0	0	0	0	0	0	shift (A) left
0	0	0	0	0	0	0	0	shift (A) left
0	0	0	0	0	0	1	1	add D(X) to (A)
0	0	0	0	0	0	1	1	
0	0	0	0	0	1	1	0	shift (A) left
0	0	0	0	1	1	0	0	shift (A) left
0	0	0	0	0	0	1	1	add D(X) to (A)
0	0	0	0	1	1	1	1	final content of A (product = decimal 15)

The procedures above form the product of the actual binary numbers in Q and X. If the multiplier and multiplicand are positive, the product formed by the process, (Q)(X), will be the desired product. If the multiplier, M_q, is negative, the content of Q is $2^{36} - 1 - |M_q|$, ahd the product formed by the computer is $2^{36}(X) - (X) - |M_q|(X)$. Since the product desired is $-|M_q|(X)$, a correction of the product formed by the computer is necessary. The correction, $-2^{36}(X) + (X)$, is made during the Multiply Sequence by (1), subtracting D(X) from A before the first performance of "Shift A left one place" (providing the correction $-2^{36}(X)$ since a shift left of 36 places follows the subtraction); and (2), adding D(X) to (A) after the last performance of "Shift Q left one place", providing the correction +(X).

An example of multiplication with a negative multiplier is given below.

(X) = 0011, multiplicand of decimal 3

(Q) = 1010, multiplicand of decimal -5

$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	Stages of A
0 0 0 0	0 0 0 0	initial content of A
0 0 0 0	0 0 1 1	subtract D(X) from (A)
1 1 1 1	1 1 0 0	
1 1 1 1	1 0 0 1	shift (A) left
0 0 0 0	0 0 1 1	add D(X) to (A)
1 1 1 1	1 1 0 0	
1 1 1 1	1 0 0 1	shift (A) left
1 1 1 1	0 0 1 1	shift (A) left
0 0 0 0	0 0 1 1	add D(X) to (A)
1 1 1 1	0 1 1 0	
1 1 1 0	1 1 0 1	shift (A) left
0 0 0 0	0 0 1 1	add D(X) to (A)
1 1 1 1	0 0 0 0	final content of A (product = -15)

A multiplication with a negative multiplicand requires no correction. A multiplication with both multiplier and multiplicand negative requires the same correction cited above.

If the instruction being executed is such that the product of (Q) and (X) is to be added to a number already in the Accumulator, the content of A is shifted 36 places to the left preceding any of the multiply operations. This positions the most significant digits of (A) in A_R in readiness for the additions of the multiplicand to A and the shifting operations. Prior to the actual multiplication operations of the Multiply Sequence, the initial content of A is tested for the condition $2^{71} > |(A)_i| \geq 2^{70}$. If this is evidenced by $(A)_{35} \neq (A)_{34}$, an A Fault is indicated on the Supervisory Control Panel, and machine operations are stopped. This condition indicates the possibility of an "overflow" during the Multiply Sequence, i.e., the modular sums resulting from the additions of D(X) to (A) may reach the positive or negative value capacity of A, $2^{71} - 1$ or $1 - 2^{71}$; and, as the result of continuing additions, become a number, $S \geq 1 - 2^{71}$ or $S \leq 2^{71} - 1$, thus destroying the cumulative effect desired.

d. DIVIDE SEQUENCE. - The machine process of division, as ordered by the Divide instruction, employs the Accumulator, the X Register, and the Q Register in such a manner that

$$(A)_i = (X) \cdot (Q) + (A)_f, \quad 0 \leq (A)_f < |(X)|.$$

The 72-bit dividend is initially contained in the Accumulator; the divisor is placed in the X Register, and the quotient is formed in the Q Register with the remainder of the division left in A. The division process utilizes the shifting facilities of Q and A and the operations which add and subtract D(X) into A. In general, the steps of the Divide Sequence are as follows.

1. Shift (A) left 36 places.

Repeat the following steps 36 times:

2. Shift (A) left once

3. Set (Q_0) to zero or one

4. Add or subtract D(X) into A

5. Shift Q left one.

The operation performed in steps 3 and 4 is determined by a relationship between appropriate digits of the dividend and divisor.

The basic principle of the Divide Sequence is as follows: decrease a postive dividend by the product of the divisor and descending powers of two, beginning with 2^{35}, until a negative number results; increase this remainder by the product of the divisor and successive descending powers of two until a positive number results; decrease this number, etc. Continue this procedure until the product of the divisor and 2^0 has been added or subtracted, yielding the final remainder. Note that the product mentioned above may in itself be positive or negative depending on the sign of the divisor. Each time a product is subtracted, a one is inserted in the right-most stage of the Q Register and subsequently shifted left once; and each time a product is added, the contents of Q, as it stands, are shifted left once. After the final shifting of those bits inserted in Q, the register will contain a correct value of the quotient although a negative final remainder as derived above may necessitate an increase or decrease in the value of the quotient. If the final remainder of the above procedure is negative, the remainder is increased by the absolute value of the divisor and the quotient adjusted accordingly by increasing or decreasing its value by one.

The examples below illustrate the basic principle of the Divide Sequence. The examples use four bit Q and X registers and an eight bit Accumulator. The machine processes of shifting the bits in the Accumulator and the Q Register, which facilitate the division machinewise, are not shown. Also, the machinewise formation of sums and differences by complementation and subtraction is not shown.

Example 1

$$\begin{array}{ll}
\text{Dividend is } 0000\ 1110,\ (A)_i & (=14)\\
\text{Divisor is } 0100,\ (X) & (=4)
\end{array}$$

$$\begin{array}{llr}
0000\ 1110 & \text{Dividend} = & +14\\
\underline{\ 010\ 0000} & \text{minus } D(X)\cdot 2^3 & \underline{-(+32)}\\
1110\ 1101 & & -18\\
\underline{0001\ 0000} & \text{plus } D(X)\cdot 2^2 & \underline{+(+16)}\\
1111\ 1101 & & -2\\
\underline{0000\ 1000} & \text{plus } D(X)\cdot 2^1 & \underline{+(+8)}\\
0000\ 0110 & & +6\\
\underline{0000\ 0100} & \text{minus } D(X)\cdot 2^0 & \underline{-(+4)}\\
0000\ 0010 & \text{Remainder} = & +2
\end{array}$$

In this example the quotient is derived as follows: the divisor is such that it may be subtracted from the dividend $2^3-2^2-2^1+2^0$ times, or in binary

$$\begin{array}{rl}
1000 & \\
-0100 & \\
-0010 & \\
\underline{+0001} & \\
0011 & (=3)
\end{array}$$

which is the content of the Q Register after the final shifting of the bits inserted into it. Thus, the final results of this division are

$$\begin{array}{ll}
\text{Quotient is } 0011,\ (Q)_f & (=3)\\
\text{Remainder is } 0000\ 0010,\ (A)_f & (=2)
\end{array}$$

Example 2

$$\begin{array}{ll}
\text{Dividend is } 0000\ 1110,\ (A)_i & (=14)\\
\text{Divisor is } 1100,\ (X) & (=-3)
\end{array}$$

$$\begin{array}{llr}
0000\ 1110 & \text{Dividend} = & +14\\
\underline{1110\ 0111} & \text{plus } D(X)\cdot 2^3 & \underline{+(-24)}\\
1111\ 0101 & & -10\\
\underline{1111\ 0011} & \text{minus } D(X)\cdot 2^2 & \underline{-(-12)}\\
0000\ 0010 & & +2\\
\underline{1111\ 1001} & \text{plus } D(X)\cdot 2^1 & \underline{+(-6)}\\
1111\ 1011 & & -4\\
\underline{1111\ 1100} & \text{minus } D(X)\cdot 2^0 & \underline{-(-3)}\\
1111\ 1110 & \text{Remainder} = & -1
\end{array}$$

In this example the quotient is derived as follows: the divisor is such that it may be subtracted from the dividend $-2^3+2^2-2^1+2^0$ times, or in binary

$$\begin{array}{rl}
-1000 & \\
+0100 & \\
-0010 & \\
\underline{+0001} & \\
1010 & (=-5)
\end{array}$$

which is the content of the Q Register after the final shifting of the bits inserted into it. Thus, the results of the division process thus far are

Quotient is 1010, (Q)	(=-5)
Remainder is 1111 1110, (A)	(=-1)

but, since the remainder resulting from this division is negative, the value of the divisor is subtracted from it, and the quotient is subsequently increased by a value of one. Thus, the final results of the division process are a

Quotient of 1011, $(Q)_f$	(=-4)
Remainder of 0000 0010, $(A)_f$	(=+2)

If the dividend is negative, the procedure for dividing is essentially the same, with the dividend being first increased to a positive number, then decreased, etc.

During the division sequence (but before the quotient is adjusted for a negative remainder, if such is the case) machine divide checks are made which determine if the value of the quotient should be an integer which would exceed the capacity, $2^{35} - 1 \leq I \leq 1 - 2^{25}$, of the Q Register. If such is the case, an A Fault is indicated on the Supervisory Control Panel, and the machine operations are stopped.

Since these divide checks are made before any final adjustments to a (negative) remainder and quotient, the division process as illustrated previously, but with a negative dividend, could result in an inaccurate value for the quotient if a negative remainder was left during a division in which the quotient was 2^{35}-1 or 1-2^{35}. To take care of such cases in such a way that the division will be stopped by a divide check, an initial correction and an end correction are made during all divisions with negative dividends. These are illustrated by the following example in which the division without the initial correction (which decreases the dividend by the value of the divisor) would result in an inaccurate quotient. The division in the example below would not be carried to completion but would be stopped, as is shown, by the occurrence of the divide check A-Fault.

Example 3

Dividend is 1110 1001, $(A)_i$	(=-22)
Divisor is 1100, (X)	(=-3)

1110 1001	Dividend	-22
1111 1100	plus D(X), initial correction	+(- 3)
1110 0110		-25
1110 0111	minus $D(X) \cdot 2^3$	(-24)
1111 1110		- 1
1111 0011	minus $D(X) \cdot 2^2$	-(-12)
0000 1011		+11

In this division the process would be stopped, after the subtraction of $D(X) \cdot 2^2$ from the previous remainder, since the quotient derived thus far would indicate that the divisor can be subtracted from the dividend $2^3 + 2^2 + (\pm 2^1 \pm 2^0)$ times, the value of which in all cases exceeds the value possible to the quotient.

If this division were performed by the Divide Sequence without making the initial correction above, it would proceed as follows:

```
1110 1001     Dividend                -22
1110 0111     minus D(X) ·2^3       -(-24)
0000 0010                              + 2
1111 0011     plus D(X) · 2^2       +(-12)
1111 0101                              -10
1111 1001     minus D(X)· 2^1       -(- 6)
1111 1011                              - 4
1111 1100     minus D(X)· 2^0       -(- 3)
1111 1110                              - 1 .
```

Thus, the division indicates that the divisor can be subtracted $+2^3-2^2+2^1+2^0$ (=7) times from the dividend leaving a remainder of -1. The correction to the quotient for this negative remainder would be

```
      0111  Quotient, (Q)          (=7)
plus     1
      1000  Incorrect value of quotient (=-7) .
```

Thus, the final contents of the Q Register would be the number above with no indication of the overflow of the modular value allowable to the quotient.

REPERTOIRE OF INSTRUCTIONS

1. GENERAL

The logic of the Univac Scientific computer is specified as a two-address logic. This means that two references are provided for the execution of an operation, and an instruction to perform this operation is coded accordingly. The references may be the addresses of operands or other instructions in storage, the address at which a result is to be stored, or they may have a special designation as noted subsequently.

An instruction word in the computer consists of 36 bits, $i_{35} \ldots i_0$, with the following composition:

Operation code, O.C.	$i_{35} \ldots i_{30}$, six bits
First execution address, u	$i_{29} \ldots i_{15}$, 15 bits
Second execution address, v	$i_{14} \ldots i_0$, 15 bits .

A programmed instruction is coded using octal notation. It is represented by octal digits as follows:

·O.C.	$(i_{35} \ldots i_{30})$	two octal digits
u	$(i_{29} \ldots i_{15})$	five octal digits representing the bits $u_{14} \ldots u_0$
v	$(i_{14} \ldots i_0)$	five octal digits representing the bits $v_{14} \ldots v_0$

According to the function of the instruction, the portions u and v of the word, represented in octal notation, may be designated as follows:

j	one-digit octal number as represented by the left-most binary digits of u, u_{14}, u_{13}, u_{12}
n	four-digit octal number as represented by the binary digits u_{11}, u_{10}, $\ldots$ u_0
k	number of shifts as represented by the right-most binary digits of v, v_6, v_5, $\ldots$ v_0; or in one case, as represented by the right-most binary digits of u, u_6, u_5, $\ldots$, u_0.

The functions of the instructions in which j, n, and k occur in place of a storage address will explain their purpose.

Following are other symbols used in the explanation of the functions of the instructions.

a. Parentheses are used to denote content(s) of, thus:

(u) = 36-bit word at address u
(Q) = 36-bit word in Q
(A) = 72-bit word in A
(A_R) = 36-bit word in A_R
(A_L) = 36-bit word in A_L

b. Lower case letters:

q_n is the bit represented by the stage Q_n, $35 \geq n \geq 0$

a_n is the bit represented by the stage A_n, $71 \geq n \geq 0$

c. A prime denotes a complement; such as (Q)' is the complement of the 36-bit word in Q.

d. Double length extensions:

D(u) is a 72-bit word whose right-hand 36 bits are (u) and whose left-hand 36 bits are all alike and equal to the left-most bit of (u).

S(u) is a 72-bit word whose right-hand 36 bits are (u) and whose left-hand 36 bits are all zeros.

D(Q), D(X), S(Q), and S(X) are similarly defined.

L(Q)(u) is a 72-bit word whose left-hand 36 bits are zeros and each of whose right-hand 36 bits is determined by the bit-by-bit product of the corresponding bits of (u) and (Q).

L(Q)'(v) is a 72-bit word whose left-hand 36 bits are zeros and each of whose right-hand 36 bits is determined by the bit-by-bit product of the corresponding bits of (v) and the complement of (Q).

2. PRESENTATION OF INSTRUCTIONS.

The listing of the Univac Scientific repertoire of instructions which follows has the instructions grouped according to a basic similarity in their functions or operations. The similarity may be due to the use which is made of the u and v address portions of the instruction, or it may be because a group of instructions uses complex internal sequences. The instructions are presented with their functions, octal codes, and the alphabetic notations which represent them.

a. TRANSMISSIVE INSTRUCTIONS. - This group of instructions uses the u address portion of the instruction as an "acquisition" address (with one exception), which designates the location from which information is to be obtained. This information may then be involved in some sort of operation, depending upon the function of the particular instruction. The v address portion of the instruction specifies the location to which the information from u, or the information resulting from any operation executed, is transmitted.

The original content of the location specified by u is not disturbed in any way by the acquisition of information from the location. The original content of the location specified by v is replaced by the information transmitted to it.

The instructions are as follows.

11uv: TRANSMIT POSITIVE (TPuv): Replace (v) with (u).

12uv: TRANSMIT MAGNITUDE (TMuv): Replace (v) with the absolute magnitude of (u).

13uv: TRANSMIT NEGATIVE (TNuv): Replace (v) with the complement of (u).

15uv: TRANSMIT U ADDRESS (TUuv): Replace the 15 bits of (v) designated by $(v_{29} \ldots v_{15})$ with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

16uv: TRANSMIT V ADDRESS (TVuv): Replace the right-hand 15 bits of (v) designated by $(v_{14} \ldots v_0)$ with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

22jkv: LEFT TRANSMIT (LTjkv): Left circular shift (A) by k places, k being $u_6 \ldots u_0$. Then replace (v) with (A_L) if j = 0, or replace (v) with (A_R) if j = 1.

35uv: ADD AND TRANSMIT (ATuv): Add D(u) to (A). Then replace (v) with (A_R).

36uv: SUBTRACT AND TRANSMIT (STuv): Subtract D(u) from (A). Then replace (v) with (A_R).

The first five instructions listed, TPuv, TMuv, TNuv, TUuv, and TVuv, do not involve any of the arithmetic registers except the X Register in their execution (unless u or v is A or Q). The information is acquired from u and placed in X where it may or may not be manipulated according to the function of the instruction. The desired content of X is then transmitted to the v-addressed location. The instructions TUuv and TVuv are intended for the modification of the u and v address portions of other instructions in storage.

The remaining three instructions in the list, LTjkv, ATuv, STuv, involve the Accumulator in their execution whether it is addressed or not. The Left Transmit instruction provides the only means of coding a transmission directly from A_L. Any other transmissions made from A to a 36-bit storage location take the content of the 36-right-most stages of A, (A_R).

The sum or difference formed in the Accumulator by ATuv or STuv is not disturbed by the transmission of (A_R) to v. The 36-bit number stored in v may be interpreted as a different quantity than the sum or difference left in A if "overflow" occurred during the addition or subtraction into the sign-bit stage A_{35}. The quantity in A must be $\leq 2^{35}-1$ in absolute value if it is to reflect the sum or difference correctly in a 36-bit storage location. If the sum or difference is used directly from A in further operations, an overflow past the stage A_{34} need not be disturbing. In this case, a check should be made for an overflow past the stage A_{70}, i.e., the sum should be $\leq 2^{71}-1$ in absolute value.

b. REPLACE INSTRUCTIONS. - This group of instructions uses the u-address portion of the instruction as an acquisition address and later places information back at the u-addressed location, destroying its original content. The v-addressed portions of the instruction are used to reference operands or hold a shift count.

The instructions are as follows.

21uv: REPLACE ADD (RAuv): Form in A the sum of D(u) and D(v). Then replace (u) with (A_R).

23uv: REPLACE SUBTRACT (RSuv): Form in A the difference D(u) minus D(v). Then replace (u) with (A_R).

27uv: CONTROLLED COMPLEMENT (CCuv): Replace (A_R) with (u) leaving (A_L) undisturbed. Then complement those bits of (A_R) that correspond to ones in (v). Then replace (u) with (A_R).

54uk: LEFT SHIFT IN A (LAuk): Replace (A) with D(u). Then left circular shift (A) by k places and replace (u) with (A_R). If u is the address of the Accumulator, the first step is omitted, so that the initial content of A is shifted.

55uk: LEFT SHIFT IN Q (LQuk): Replace (Q) with (u). Then left circular shift (Q) by k places and replace (u) with (Q).

The content of A is left undisturbed by the transmission of (A_R) to u as effected by the first four instructions above. The content of Q is undisturbed by the transmission to u during the Left Shift in Q instruction.

The sum or difference formed by RAuv or RSuv must be $\leq 2^{35}-1$ in absolute value if the quantity stored at a 36-bit location u is to reflect the proper value.

Note that the function of instruction CCuv may be thought of as "Replace the content of u with the bit-by-bit sum of (u) and (v), disregarding any carries propagated".

c. SPLIT INSTRUCTIONS. - The following group of instructions uses the u address of the instruction as an acquisition address and assumes in the X Register the split extension of the content of the specified location. S(X) is then added or subtracted into the accumulator after which the content of A may be shifted. These instructions make it possible to effect a change in (A_R) without disturbing (A_L).

The instructions are as follows.

31uk: SPLIT POSITIVE ENTRY (SPuk): Form S(U) in A. Then left circular shift (A) by k places, k being $v_6...v_0$ and $v_{14}...v_7$ being zero.

32uk: SPLIT ADD (SAuk): Add S(u) to (A). Then left circular shift (A) by k places, k being $v_6...v_0$ and $v_{14}...v_7$ being zero.

33uk: SPLIT NEGATIVE ENTRY (SNuk): Form in A the complement of S(u). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

34uk: SPLIT SUBTRACT (SSuk): Subtract S(u) from (A). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

d. Q-CONTROLLED INSTRUCTIONS. - These instructions use the u address as an acquisition address and transmit to the v-addressed location the final content of A_R. The final content of A_R is effected by the content of the Q Register which is used as a "mask". The "l's" in the information acquired from the u-addressed location are retained only when there are "l's" in the corresponding stages of Q.

The instructions are:

51uv: Q-CONTROLLED TRANSMIT (QTuv): Form in A the number L(Q)(u). Then replace (v) with (A_R).

52uv; Q-CONTROLLED ADD (QAuv): Add to (A) the number L(Q)(u). Then replace (v) with (A_R).

53uv: Q-CONTROLLED SUBSTITUTE (QSuv): Form in Q the quantity L(Q)(u) plus L(Q)'(v). Then replace (v) with (A_R). This effects the replacement of selected bits of (v) with the corresponding bits of (u) in those places corresponding to one's in Q.

e. SEQUENCED INSTRUCTIONS. - The following instructions use complex sequences of computer operations.

71uv: MULTIPLY (MPuv): Form in A the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

72uv: MULTIPLY ADD (MAuv): Add to (A) the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

73uv: DIVIDE (DVuv): Divide the 72-bit number in A by (u), putting the quotient in Q, and leaving in A a non-negative remainder R. Then replace (v) by (Q). The quotient and remainder are defined by: $(A)_i = (u) \cdot (Q) + R$, where $0 \leq R < |(u)|$. $(A)_i$ denotes the initial content of A.

74uv: SCALE FACTOR (SFuv); Replace (A) with D(u) (unless u is A). Then left circular shift (A) by 36 places and continue shifting until $a_{34} \neq a_{35}$. Replace the right-hand 15 bits of (v) with the number of left circular shifts, k, which would be necessary to return (A) to its original position.

75jnw: REPEAT (RPjnw): This instruction calls for the next instruction, which will be called NIuv, to be executed n times, its "u" and "v" addresses being modified or not according to the value of j. Normally n executions are made and the program is continued by the execution of the instruction stored at a fixed address F_1, 00000 or 40001. The procedure is:

(1) Replace the right-hand 15 bits of (F_1) with the address w

(2) Execute NIuv, the next instruction in the program n times

(3) If j = 0, do not change u and v
If j = 1, add one to v after each execution
If j = 2, add one to u after each execution
If j = 3, add one to u and v after each execution.

(4) On completing n executions, take (F_1) as the next instruction.

(5) If the repeated instruction is a jump or stop instruction, the occurrence of a jump or stop terminates the repetition. In addition, if NIuv is a Threshold Jump or an Equality Jump, and the jump to address v occurs, (Q) is replaced by the quantity, j, n-r where r is the number of executions that has taken place.

The instructions MPuv and MAuv use the Multiply Sequence explained in the section General Description, Arithmetic Operations. Note the overflow indication feature of the Multiply Add instruction. If an overflow (into the sign-bit stage, A_{71}) of the sum of the product of u and v and the current content of A is possible, a computer fault is incurred. The computer fault is indicated by the illumination of a light on the Supervisory Control Panel.

The Divide instruction uses the Divide Sequence, also explained in the section General Description, Arithmetic Operations.

The Scale Factor instruction is useful in "normalizing" a number, i.e., scaling a number to its maximum representation in 36 stages so that the most significant bit is held in the stage adjacent to the stage holding the sign bit.

The Repeat instruction is an extremely useful feature of the computer. Consider, for example, its effectiveness in performing a "block" transfer, i.e., the transfer of a group of words from their storage at one set of consecutive locations to another set of consecutive locations. The instruction Transmit Positive (TPuv) causes the word at address u to be transmitted to address v. The simple two-instruction routine

$$\left.\begin{array}{lll} \text{RP} & 3,n & w \\ \text{TP} & u_1 & v_1 \end{array}\right\} \quad \text{Transfer } n \text{ words } (u_1) \rightarrow v_1, (u_2) \rightarrow v_2, \ldots, (u_n) \rightarrow v_n$$

effects a block transfer of n words from register u_1 through u_n to registers v_1 through v_n. If the transmission are from registers u_i in Magnetic Drum Storage to registers v_i in Rapid Access Storage, the transfer rate is about 31,000 words per second.

Another example of the effectiveness of the Repeat instruction is provided by its use preceding a Multiply Add instruction to form the product accumulation of $a_1 b_1 + a_2 b_2 + a_3 b_3 + a_n b_n$.

f. ONE-WAY CONDITIONAL JUMP INSTRUCTIONS. - These instructions acquire a word from the location specified by the u address and, if a condition involving this word is satisfied, effect a jump to the instruction at the location specified by the v address.

The instructions are as follows.

41uv: INDEX JUMP (IJuv): Form in A the difference D(u) minus one. Then if a_{71} is one, continue the present sequence of instructions; if a_{71} is zero, replace (u) with (A_R) and take (v) as NI.

42uv: THRESHOLD JUMP (TJuv): If D(u) is greater than (A), take (v) as NI; if not, continue the present sequence. In either case, leave (A) in its initial state.

43uv: EQUALITY JUMP (EJuv): If D(u) equals (A), take (v) as NI; if not, continue the present sequence. In either case leave (A) in its initial state.

Note that a positive quantity acquired from u by the Index Jump instruction, and decreased by the value of one, is returned to storage in u until the quantity becomes negative. This provides a means of "counting", and (u) is sometimes referred to as a "counter". This feature is extremely useful in performing an iterative cycle which is to be repeated a prescribed number of times.

The instructions TJuv and EJuv are made more useful by the feature which leaves the content of A in its initial state after the comparison tests are made.

g. TWO-WAY CONDITIONAL JUMP INSTRUCTIONS. - These instructions use both the u- and v-addressed portions of the instruction to specify an instruction to be executed next. The direction of the jump depends on the condition of a quantity which is tested in the Accumulator or the Q Register. Since either the u and v address is available to be used as an acquisition address, the quantity must be placed in A or Q previous to the execution of one of these jump instructions.

These instructions are ts follows.

44uv: Q JUMP (QJuv): If q_{35} is one, take (u) as NI; if q_{35} is zero, take (v) as NI. Then, in either case, left circular shift (Q) by one place.

46uv: SIGN JUMP (SJuv): If a_{71} is one, take (u) as NI; if a_{71} is zero, take (v) as NI.

47uv: ZERO JUMP (ZJuv): If (A) is not zero, take (u) as NI; if (A) is zero, take (v) as NI.

Note that a zero condition (of q_{35}, a_{71}, or the content of A) always causes a jump to the v-addressed instruction.

h. ONE-WAY UNCONDITIONAL JUMP INSTRUCTIONS. - This group of instructions does not depend upon some condition being satisfied by a machine word to cause a jump.

The instructions are as follows.

14--: INTERPRET (IP--): Let Y represent the address from which CI was obtained. Replace the right-hand 15 bits of (F_1) with the quantity Y + 1. Then take (F_2) as NI.

37uv: RETURN JUMP (RJuv): Let y represent the address from which CI was obtained. Replace the right-hand 15 bits of (u) with the quantity y + 1. Then take (v) as NI.

45jv: MANUALLY SELECTIVE JUMP (MJjv): If the number j is 0, take (v) as NI. If j is 1, 2, or 3, and the correspondingly numbered MJ selecting switch on the control panel is set to "jump", take (v) as NI; if this switch is not set to "jump", continue the present sequence.

Note that the jump instituted by the Manual Jump instruction with a j of 1, 2, or 3 is conditional with regard to a manual selection.

The Return Jump instruction provides a means of (1) interrupting the sequence of instructions being executed currently and (2) returning to this sequence after jumping out of it. This is effected if the instruction at u is a Manual Jump instruction, and it is executed at some time following the execution of the instruction at v.

The execution of the Interpret instruction is equivalent to executing a Return Jump with u = F_1 and v = F_2.

i. EXTERNAL EQUIPMENT INSTRUCTIONS. - The following instructions provide for input to and output from the computer. Input information is transmitted to the location specified by the v address of an input instruction; during output operations, the information to be transmitted to external equipment is acquired from the v-addressed location. The use of these instructions is explained in detail in the section of this volume, Input and Output Systems.

The instructions are as follows.

61-v: PRINT (PR-v): Replace (TWR) with the right-hand six bits of (v). Cause the typewriter to perform the operation specified by the 6-bit code.

63jv: PUNCH (PUjv): Replace (HPR) with the right-hand six bits of (v). Cause the punch to respond to (HPR). If j = 0, omit seventh level hole, if j = 1, include seventh level hole.

17-v: EXTERNAL FUNCTION (EF-v): As indicated by (v) select a unit of external equipment and instruct it to perform the designated function.

76jv: EXTERNAL READ (ERjv): If j = 0, replace (v) with (IOA) in $v_7 \ldots v_0$ and zeros in $v_{35} \ldots v_8$; if j = 1, replace (v) with (IOB).

77jv: EXTERNAL WRITE (EWjv): If j = 0, replace (IOA) with the right-hand eight bits of (v), if j = 1, replace (IOB) with (v).

j. STOP INSTRUCTIONS. - The following instructions cause a stop of computer operation and are self-explanatory:

56jv: MANUALLY SELECTIVE STOP (MSjv): If j is 0, stop computer operation, indicating the stop by the illumination of a light on the Supervisory Control Panel. If j is 1, 2, or 3 and the correspondingly numbered MS selecting switch is set to "stop", stop computer operation, indicating the stop by the illumination of a light on the Supervisory Control Panel. Whether or not a stop occurs, (v) is NI.

57--: PROGRAM STOP (PS--): Stop computer operation, indicating the stop by the illumination of a light on the Supervisory Control Panel.

SEQUENTIAL PRESENTATION OF INSTRUCTIONS

In the following pages the instructions are presented with their representative octal operation codes in numerical order. The purpose of this presentation is to give the programmer a simplified representation of the detailed operations which the computer performs during, and necessary to, the execution of each instruction. The events listed under each instruction occur as a result of the control section receiving and sensing the operation code of the instruction. These events do not necessarily designate a machine operation but represent the actions taken by the machine in executing an instruction. The order in which these events are listed is in accordance, for the most part, with the time-wise sequence of machine operations which they represent. In some instructions the desire for a simplified presentation overruled the desire for a sequential presentation if such was not necessary for accuracy in determining the contents of the various registers at any time during the execution of the instruction. The deviations from machine operation sequences in any of the presentations do not cause a misrepresentation of the actual operations of the machine.

The columns in which the events are listed differentiate the events in accordance with their primary function in the execution of the instruction. An event listed in the Procurement of Operands column brings an operand out of storage (MC, MD, Q, or A) into the X Register in preparation for some operation upon it or placement in another register. An event listed in the Operations column may describe arithmetic and/or logical machine operations or may be the clearance of a register in preparation for such operations. An event listed in the third column, Storage of Results, indicates the final placement of a result of actions described in the first and/or second columns.

The symbols and abbreviations used in the presentation of the instructions have been listed previously except for those listed below. The definitions of the following notations do not explain machine operations but the effect of machine operations.

Clear A	Replace the contents of each stage of the Accumulator with a zero.
Clear X	Replace the contents of each stage of the X Register with a zero.
Complement PAK	Replace the contents of each stage of PAK with its complement.
Complement X	Replace the contents of each stage of X with its complement.
$\left.\begin{matrix}u\\v\end{matrix}\right\}\longrightarrow$ PAK	Replace the contents of the 15 stages of the Program Address Counter with the MC or MD storage address as designated by u or v.
$F_1 \longrightarrow$ PAK	A special case of the above with the MC address specified as being octal 00000, (or MD address 40001).
jn $\longrightarrow$ PAK	Replace the contents of the 15 stages of PAK with the bits designated as jn.
$(X_{14}\ldots X_0)\rightarrow$PAK	Replace the contents of the 15 stages of PAK with the contents of the right-most stages of the X Register.
PAK $\longrightarrow$ X	Replace the contents of the 15 right-most stages of the X Register $(X_{14}\ldots X_0)$, with the address held in PAK.

$w \longrightarrow X$	Replace the contents of the 15 right-most stages of the X Register, $(X_{14}...X_0)$, with the address designated as w.
$k \longrightarrow X$	Replace the contents of the right-most stages of the X Register with the shift count k.
$(IOA) \rightarrow X$	Replace the contents of the eight right-most stages of X with (IOA).
$(IOB) \rightarrow X$	Replace the contents of the 36 stages of X with (IOB)
$(PAK) \rightarrow X$	Replace the contents of the 36 stages of X with the contents of the address held in PAK.
$(u) \rightarrow X$	If u is an MC or MD address, replace the contents of the X Register with the contents of the MC or MD location specified by u. If u is an address of the Q Register, replace the contents of the X Register with the contents of the Q Register. If u is an address of the Accumulator, replace the contents of the X Register with the contents of A_R lnless otherwise noted.
$(v) \rightarrow X$	If v is the address of an MC or MD location or Q or A, replacement operations occur as above with v being the transmitting register.
$(X) \rightarrow IOA$	Replace the contents of the eight stages of IOA with the contents of the right-most eight stages of the X Register.
$(X) \rightarrow IOB$	Replace the contents of the 36 stages of IOB with the contents of the X Register.
$(X) \rightarrow PCR$	Replace the contents of the 36 stages of PCR with the contents of the X Register.
$(X) \longrightarrow u$	If u is an MC or MD address, replace the contents of the MC or MD location specified by u with the contents of the X Register. If u is an address of the Q Register, replace the contents of Q with the contents of the X Register.
$(X) \longrightarrow v$	If u or v is an address of the Accumulator, replacement operations do not occur except in instructions 11, 12, 13, 22, 55, 73, and 76. In these instructions, the replacement is performed by Clear A and Add D(X) to A. A replacement operation $(X) \longrightarrow A$ occurring in the instructions not listed above would effect undesirable alterations in the contents of A. If v is the address of an MC or MD location or Q or A, replacement operations occur as above with v being the receiving register.
$(Xn_2...Xn_1) \longrightarrow \begin{cases} u \\ v \end{cases}$	If the receiving register has an MC or MD address, replace with the contents of the stages $Xn_2...Xn_1$ only the contents of the corresponding stages of the receiving register, leaving the remaining stages undisturbed. A partial replacement of Q or A is not permissible.
$(X_{14}...X_0) \rightarrow F_1$	A special case of the above with the stages of X being specifiel as being the right-most 15 and the MC address specified as being 00000 (or MD address 40001).

The contents of the transmitting register are not disturbed in any of the above replacement operations.

It should be remembered that the Add D(X) to (A) and Add S(X) to (A) operations cited are performed machine-wise by complementation and subtraction.

Instruction Reference Events

The events listed below normally conclude the execution of all computer instructions, except the Repeat instruction, when they are not immediately preceded by the Repeat instruction. Also, a number of instructions which are preceded by the Repeat instruction are terminated by these events. (Refer to the Repeat instruction and the Termination of a Repeat Sequence.) It is understood that these events follow in sequential order the events listed for each appropriate instruction. As a result, the execution of the current instruction is terminated, and the computer is prepared for the execution of the next instruction whose address is acquired from PAK, the Program Address Counter. This address is determined according to the function of the current instruction if this function indicates that the normal sequence of instructions be interrupted. The contents of the location whose address is held in PAK is placed in PCR, the Program Control Register, where it is interpreted as the next instruction to be executed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	$(PAK) \longrightarrow X$ Advance PAK by one $(X) \longrightarrow PCR$	

Interrupt Termination Sequence

The Program Interrupt feature of the Univac Scientific provides a means of interrupting the usual acquisition of instructions as depicted above. When the "interrupt" becomes effective, the first two events listed above do not occur; instead the first instruction which is normally terminated by the instruction reference events above is terminated by the following events:

$$(00002) \longrightarrow X$$

$$(X) \longrightarrow PCR.$$

These steps set the instruction at F_3 (00002) into PCR as the next instruction to be executed and leave the address of the instruction normally executed next in PAK. If the instruction at F_3 is a Return Jump instruction, the address at PAK is stored at some appropriate computer location u before the jump to the v address of RJuv is made.

If the Repeat instruction, or an instruction preceded by the Repeat instruction, is being executed when the "interrupt" is selected, the events above terminate the instruction stored at F_1 unless a jump (or stop) is called for during the Repeat Sequence. If this is the case, the events above terminate the jump instruction, and the address to which the jump was to be made remains in PAK.

OPERATION CODE: 11

INSTRUCTION: Transmit Positive, TPuv

FUNCTION: Replace (v) with (u)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$		$(X) \rightarrow v$ If v is A, Clear A Add D(X) to (A)

OPERATION CODE: 12

INSTRUCTION: Transmit Absolute Magnitude, TMuv

FUNCTION: Replace (v) with the absolute magnitude of (u)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Complement (X) if negative	$(X) \rightarrow v$ If v is A, Clear A Add D(X) to (A)

OPERATION CODE: 13

INSTRUCTION: Transmit Negative, TNuv

FUNCTION: Replace (v) with the complement of (u)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Complement (X)	$(x) \rightarrow v$ If v is A, Clear A Add D(X) to (A)

OPERATION CODE: 14

INSTRUCTION: Interpret, IP

FUNCTION: Let Y represent the address from which CI was obtained. Replace the right-hand 15 bits of (F_1) with the quantity Y + 1. Then take (F_2) as the next instruction.

F_1 is storage address C0000, F_2 is storage address 00001.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
	PAK (i.e., Y + 1)$\rightarrow$X	
		$(X_{14}...X_0)\rightarrow F_1$
	Set PAK to F_2	

INSTRUCTION: Transmit U Address, TUuv

FUNCTION: Replace the 15 bits of (v) designated by (v_{29}... v_{15}) with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		$(X_{29} \ldots X_{15}) \longrightarrow v$ v = Q or A not permissible

OPERATION CODE: 16

INSTRUCTION: Transmit V Address, TVuv

FUNCTION: Replace the right-hand 15 bits of (v) designated by (v_{14}... v_0) with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$		$(X_{14}...X_0) \rightarrow v$ v = Q or A not permissible

OPERATION CODE: 17

INSTRUCTION: External Function, EF-v

FUNCTION: As indicated by (v) select a unit of external equipment and instruct it to perform the designated function.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(v) \rightarrow X$	If previous operations involving (IOB) are completed - $(X) \rightarrow IOB$ According to (IOB), select external equipment and instruct it to perform a function.	

OPERATION CODE: 21

INSTRUCTION: Replace Add, RAuv

FUNCTION: Form in A the sum of D(u) and D(v). Then replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) $\rightarrow$ X		
	Clear A	
	Add D(X) to (A)	
(v) $\rightarrow$ X		
	Add D(X) to (A)	
	(A_R) $\rightarrow$ X	
		(X) $\rightarrow$ u Omit if u is A

OPERATION CODE: 22

INSTRUCTION: Left Transmit, LTjkv

FUNCTION: Left circular shift (A) by k places, k being $u_6 \ldots u_0$. Then replace (v) with (A_L) if j=0, or replace (v) with (A_R) if j=1.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Shift (A) left k places If j is 0, $(A_L) \longrightarrow X$ If j is 1, $(A_R) \longrightarrow X$	$X \longrightarrow v$ If v is A Clear A Add D(X) to A

OPERATION CODE: 23

INSTRUCTION: Replace Subtract, RSuv

FUNCTION: Form in A the difference D(u) minus D(v). Then replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Clear A	
	Add D(X) to (A)	
$(v) \longrightarrow X$		
	Subtract D(X) from (A)	
	$(A_R) \longrightarrow X$	
		$(X) \longrightarrow u$ Omit if u is A

OPERATION CODE: 27

INSTRUCTION: Controlled Complement, CCuv

FUNCTION: Replace (A_R) with (u) leaving (A_L) undisturbed. Then complement those bits of (A_R) that correspond to ones in (v). Then replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u)$\longrightarrow$X		
	Clear A_R	
	Complement (X)	
	Complement (A_{35} ... A_0) if corresponding (X_{35} ... X_0) is zero	
(v)$\longrightarrow$X		
	Complement (X)	
	Complement (A_{35} ... A_0) if corresponding (X_{35} ... X_0) is zero	
	(A_R)$\longrightarrow$X	
		(X)$\longrightarrow$u Omit if u is A

OPERATION CODE: 31

INSTRUCTION: Split Positive Entry, SPuk

FUNCTION: Form S(u) in A. Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Clear A	
	Add S(X) to (A)	
	Shift (A) left k places	

If the bits $v_{14} \ldots v_7$ of instruction 31 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows.

(If instruction 31 is at address y, PAK will currently contain address y + 1.)

$$
\begin{aligned}
r_0 &= PAK_0 \\
\cdot\ &\quad \cdot \\
\cdot\ &\quad \cdot \\
\cdot\ &\quad \cdot \\
r_6 &= PAK_6
\end{aligned}
$$

$$
\left.
\begin{aligned}
r_7 &= PAK_7 + v_7 \\
\cdot\ &\quad \cdot \\
\cdot\ &\quad \cdot \\
\cdot\ &\quad \cdot \\
r_{14} &= PAK_{14} + v_{14}
\end{aligned}
\right\}
$$

bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

OPERATION CODE: 32

INSTRUCTION: Split Add, SAuk

FUNCTION: Add S(u) to (A). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Add S(X) to (A)	
	Shift (A) left k places	

If the bits $v_{14} \ldots v_7$ of instruction 32 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows.

(If instruction 32 is at address y, PAK will currently contain address y + 1.)

$$
\begin{aligned}
r_0 &= PAK_0 \\
&\vdots \\
r_6 &= PAK_6
\end{aligned}
$$

$$
\left.
\begin{aligned}
r_7 &= PAK_7 + v_7 \\
&\vdots \\
r_{14} &= PAK_{14} + v_{14}
\end{aligned}
\right\}
$$

bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

OPERATION CODE: 33

INSTRUCTION: Split Negative Entry, SNuk

FUNCTION: Form in A the complement of S(u). Then left circular shift (A) by k places, k being $v_6 \dots v_0$ and $v_{14} \dots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Clear A	
	Subtract S(X) from (A)	
	Shift (A) left by k places	

If the bits $v_{14} \dots v_7$ of instruction 33 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \dots r_0$, being determined as follows.

(If instruction 33 is at address y, PAK will currently contain address y + 1.)

$$\begin{aligned} r_0 &= PAK_0 \\ &\vdots \\ r_6 &= PAK_6 \end{aligned}$$

$$\left.\begin{aligned} r_7 &= PAK_7 + v_7 \\ &\vdots \\ r_{14} &= PAK_{14} + v_{14} \end{aligned}\right\}$$

bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

OPERATION CODE: 34

INSTRUCTION: Split Subtract, SSuk

FUNCTION: Subtract S(u) from (A). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$	Subtract S(X) from (A) Shift (A) by left by k places	

If the bits $v_{14} \ldots v_7$ of instruction 34 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows.

(If instruction 34 is at address y, PAK will currently contain address y + 1.)

$$
\begin{aligned}
r_0 &= PAK_0 \\
&\vdots \\
r_6 &= PAK_6 \\
r_7 &= PAK_7 + v_7 \\
&\vdots \\
r_{14} &= PAK_{14} + v_{14}
\end{aligned}
\Bigg\}
$$

bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

OPERATION CODE: 35

INSTRUCTION: Add and Transmit, ATuv

FUNCTION: Add D(u) to (A). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Add D(X) to (A) $(A_R) \rightarrow X$	$(X) \rightarrow v$ Omit if v is A

PX 36

OPERATION CODE: 36

INSTRUCTION: Subtract and Transmit, STuv

FUNCTION: Subtract D(u) from (A). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u)$\rightarrow$X	Subtract D(X) from (A) (A_R)$\rightarrow$X	(X)$\rightarrow$v Omit if v is A

OPERATION CODE: 37

INSTRUCTION: Return Jump, RJuv

FUNCTION: Let y represent the address from which CI was obtained. Replace the right-hand 15 bits of (u) with the quantity y plus one. Then take (v) as the next instruction. (See notes on the page following the Zero Jump instruction, operation code 47.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X PAK (i.e., y + 1)$\rightarrow$X v$\rightarrow$PAK	$(X_{14} \ldots X_0)\rightarrow u$ u = Q or A not permissible

OPERATION CODE: 41

INSTRUCTION: Index Jump, IJuv

FUNCTION: Form in A the difference D(u) minus one. Then if a_{71} is one, continue the present sequence of instructions; if a_{71} is zero, replace (u) with (A_R) and take (v) as NI. (See notes on the page following the Zero Jump instruction, operation code 47.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
(u)$\longrightarrow$X Omit if u is A		
	Clear A Omit if u is A	
	Add D(X) to (A)	
	Subtract one from (A)	
	(A_R)$\longrightarrow$X	
if a_{71} is zero	v$\longrightarrow$PAK	(X)$\longrightarrow$u Omit if u is A

OPERATION CODE: 42

INSTRUCTION: Threshold Jump, TJuv, not repeated

FUNCTION: Subtract (u) from (A). If a_{71} is then one, take (v) as the NI; if a_{71} is zero, continue the present sequence of instructions. Then in either case, restore (A) to its initial state. (See notes on the page following the Zero Jump instruction, operation code 47 .)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$		
	Subtract D(X) from (A)	
	$v \rightarrow$ PAK if a_{71} is one	
	Add D(X) to (A)	

OPERATION CODE: 42

INSTRUCTION: Threshold Jump, TJuv, repeated

FUNCTION: Subtract (u) from (A). If a_{71} is then one, replace (Q) with j, n-r and take (v) as the next instruction; if a_{71} is zero, repeat the execution. In either case, restore (A) to its initial state.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) $\rightarrow$ X	Subtract D(X) from (A) If a_{71} is zero, Add D(X) to (A) Advance (PAK) Test (PAK) for condition indicating n repeats and proceed accordingly (See Note 2). If a_{71} is one, Complement (PAK) Add D(X) to (A) Perform jump termination (See Note 2).	

Note: 1. This instruction is preceded by instruction 75jnw which leaves the complement of "jn" in PAK.

2. See Jump Termination in the discussion of the Repeat instruction, operation code 75.

OPERATION CODE: 43

INSTRUCTION: Equality Jump, EJuv, not repeated

FUNCTION: Subtract (u) from (A). If (A) is then zero, take (v) as the next instruction; if (A) is not zero, continue the present sequence. In either case, restore (A) to its initial state. (See notes on the page following the Zero Jump instruction, operation code 47

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Subtract D(X) from (A)	
	Subtract one from (A)	
	$v \longrightarrow$ PAK if end-borrow was propagated from A_{71}	
	Add D(X) to (A)	
	Set (X) to 1	
	Add D(X) to (A)	

OPERATION CODE: 43

INSTRUCTION: Equality Jump, EJuv, repeated

FUNCTION: Subtract (u) from (A). If (A) is then zero, replace (Q) with j, n-r and take (v) as next instruction; if (A) is not zero, repeat the execution. In either case, restore (A) to its initial state.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) $\rightarrow$ X		
	Subtract D(X) from (A)	
	Subtract one from (A)	
	If no end-borrow was propagated from A_{71} Add D(X) to (A) Set (X) to one Add D(X) to (A) Advance PAK Test (PAK) for condition indicating n repeats and proceed accordingly. (See Note 2.)	
	If end-borrow was propagated from A_{71} Complement (PAK) Add D(X) to (A) Set (X) to one Add D(X) to (A) Perform jump termination (See Note 2.)	

Note 1. This instruction is preceded by instruction 75jnw which leaves the complement of "jn" in PAK.

Note 2. See Jump Termination in the discussion of the Repeat instruction, operation code 75.

OPERATION CODE: 44

INSTRUCTION: Q-Jump, QJuv

FUNCTION: If q_{35} is one, take (u) as NI; if q_{35} is zero, take (v) as NI. Then, in either case, left circular shift (Q) by one place. (See notes on the page following the Zero Jump instruction, operation code 47

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	If q_{35} is one, $u \longrightarrow$ PAK If q_{35} is zero, $v \longrightarrow$ PAK Shift (Q) left one place	

OPERATION CODE: 45

INSTRUCTION: Manually Selective Jump, MJjv

FUNCTION: If the number j is 0, take (v) as NI. If j is 1, 2, or 3, and the correspondingly numbered MJ Selecting Switch is set to "jump", take (v) as the NI; if this switch is not set to "jump", continue the present sequence. (See notes on the page following the Zero Jump instruction, operation code 47.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	For jump, v $\longrightarrow$ PAK	

OPERATION CODE: 46

INSTRUCTION: Sign Jump, SJuv

FUNCTION: If a_{71} is one, take (u) as NI; if a_{71} is zero, take (v) as NI. (See notes on the page following the Zero Jump instruction, operation code 47.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	If a_{71} is zero, $v \longrightarrow$ PAK If a_{71} is one, $u \longrightarrow$ PAK	

OPERATION CODE: 47

INSTRUCTION: Zero Jump, ZJuv

FUNCTION: If (A) is not zero, take (u) as NI; if (A) is zero, take (v) as NI. (See notes on the following page.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Subtract one from (A) v $\rightarrow$ PAK if end-borrow was propagated from A_{71} u $\rightarrow$ PAK if no end-borrow was propagated from A_{71} Set (X) to one Add D(X) to (A)	

Notes Concerning the Jump Instructions

When the conditions are such that the contents of the v (or u) address are referred to as the next instruction to be executed, the following notes are applicable.

1. Machinewise, v = A is not permissible.

2. If v = Q, the next instruction to be executed after the termination of the jump instruction will be (Q). If (Q) is a legal instruction it will be executed in the normal manner. Its execution will be repeated, unless it is a jump or stop instruction or a Force stop is made, since the address of each succeeding instruction is taken from PAK, and PAK will advance from 31000 to 31777. If PAK is advanced past 31777 to 32000, an A address, operation is halted since an A address is not permissible.

3. The above remarks also apply to u for the two-way jump instructions.

OPERATION CODE: 51

INSTRUCTION: Q-Controlled Transmit, QTuv

FUNCTION: Form in A the number L(Q)(u). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Clear A	
	Form in X bit-by-bit product of (Q) and (X)	
	Add S(X) to (A)	
		$(X) \longrightarrow v$ Omit if v is A

OPERATION CODE: 52

INSTRUCTION: Q-Controlled Add, QAu.v

FUNCTION: Add to (A) the number L(Q)(u). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) $\longrightarrow$ X		
	Form in X bit-by-bit product of (Q) and (X)	
	Add S(X) to (A)	
	(A_R) $\longrightarrow$ X	
		(X) $\longrightarrow$ v Omit if v is A

OPERATION CODE: 53

INSTRUCTION: Q-Controlled Substitute, QSuv

FUNCTION: Form in A the quantity L(Q)(u) plus L(Q)'(v). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) $\rightarrow$ X		
	Clear A	
	Form in X bit-by-bit product of (Q) and (X)	
	Add S(X) to (A)	
	Complement (Q)	
(v) $\rightarrow$ X		
	Form in X bit-by-bit product of (Q) and (X)	
	Add S(X) to (A)	
	Complement (Q)	
	(A_R) $\rightarrow$ X	
		(X) $\rightarrow$ v Omit if v is A

OPERATION CODE: 54

INSTRUCTION: Left Shift in A, LAuk

FUNCTION: Replace (A) with D(u). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$, and replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
$(u) \longrightarrow X$ Omit if u is A		
	Clear A Omit if u is A	
	Add D(X) to (A)	
	Shift (A) left k places	
	$(A_R) \longrightarrow X$	
		$(X) \longrightarrow u$* Omit if u is A

*The 54 (and 55) instruction does not always store the shifted number at the address from which the number, before shifting, was obtained. The address to which the $(A_R)f$ will be transmitted is determined in this manner: If the 15 bits of the receiving address of the shifted number are designated as $r_{14} \ldots r_0$, these bits are determined as follows.

$$\begin{array}{l} r_0 = u_0 \\ \vdots \\ r_6 = u_6 \\ \left.\begin{array}{l} r_7 = u_7 + v_7 \\ \vdots \\ r_{14} = u_{14} + v_{14} \end{array}\right\} \end{array}$$

bit-by-bit sum with the exception that a u_i of "1" and a v_i of "1" will result in an r_i of "1"

Therefore, by choosing $v_{14} \ldots v_7$ with care, (u) may be left undisturbed. A particular advantage of this peculiarity is that the shifted number may be placed in Q or left in A without designating Q or A as the u address.

OPERATION CODE: 55

INSTRUCTION: Left Shift in Q, LQuk

FUNCTION: Replace (Q) with (u). Then left circular shift (Q) by k places, k being $v_6 \ldots v_0$, and replace (u) with (Q).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$		
	$(X) \rightarrow Q$	
	Shift (Q) left k places	
	$(Q) \rightarrow X$	
		$(X) \rightarrow u$* If u is A, Clear A Add D(X) to (A)

*The 55 (and 54) instruction does not always store the shifted number at the address from which the number, before shifting, was obtained. The address to which the $(A_R)_f$ will be transmitted is determined in this manner: If the 15 bits of the receiving address of the shifted number are designated as $r_{14} \ldots r_0$, these bits are determined as follows.

$$\begin{array}{l} r_0 = u_0 \\ \vdots \\ r_6 = u_6 \end{array}$$

$$\left.\begin{array}{l} r_7 = u_7 + v_7 \\ \vdots \\ r_{14} = u_{14} + v_{14} \end{array}\right\} \text{bit-by-bit sum with the exception that a } u_i \text{ of "1" and a } v_i \text{ of "1" will result in an } r_i \text{ of "1".}$$

Therefore, by choosing $v_{14} \ldots v_7$ with care, (u) may be left undisturbed. A particular advantage of this peculiarity is that the shifted number may be placed in Q or left in A without designating Q or A as the u address.

OPERATION CODE: 56

INSTRUCTION: Manually Selective Stop, MSjv

FUNCTION: If j is 0, stop the computer operation, indicating the stop by the illumination of a light on the Supervisory Control Panel. If j is 1, 2, or 3 and the correspondingly numbered MS selecting switch is set to "stop", stop computer operation, indicating the stop by the illumination of a light on the Supervisory Control Panel. Whether or not a stop occurs, (v) is NI.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	$v \longrightarrow$ PAK Stop computer operation if stop indicated.	

Note: v = A not permissible; for v = Q, see Note 2 on the page following the Zero Jump instruction, operation code 47.

OPERATION CODE: 57

INSTRUCTION: Program Stop, PS--

FUNCTION: Stop computer operation, indicating the stop by the illumination of a light on the Supervisory Control Panel.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Stop computer operation.	

OPERATION CODE: 61

INSTRUCTION: Print, PR-v

FUNCTION: Replace (TWR) with the right-hand six bits of (v). Cause the typewriter to perform the operation specified by the 6-bit code.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	$(v) \longrightarrow X$ If TWR is clear, place one's in $(TWR_5 \ldots TWR_0)$ where they are present in corresponding $(X_5 \ldots X_0)$ Perform typewriter operation as specified by (TWR).	

OPERATION CODE: 63

INSTRUCTION: Punch, PUjv

FUNCTION: Replace (HPR) with the right-hand six bits of (v). Cause the punch to respond to (HPR). If j = 0, omit seventh level hole; if j = 1 include seventh level hole.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	$(v) \longrightarrow X$ If HPR is clear, place a one in HPR_6 if j = 1 and place ones in $(HPR_5 ... HPR_0)$ where they are present in corresponding $(X_5 ... X_0)$ Punch (HPR)	

OPERATION CODE: 71

INSTRUCTION: Multiply, MPuv

FUNCTION: Form in A the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Clear A	
	$(X) \longrightarrow Q$	
$(v) \longrightarrow X$		
	Form in A the product of (Q) and (X)	

OPERATION CODE: 72

INSTRUCTION: Multiply Add, MAuv

FUNCTION: Add to (A) the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	$(X) \longrightarrow Q$	
	Shift (A) left 36 places	
$(v) \longrightarrow X$		
	Add to (A) shifted the product of (Q) and (X)	

OPERATION CODE: 73

INSTRUCTION: Divide, DVuv

FUNCTION: Divide the 72-bit number in A by (u), putting the quotient in Q, and leaving in A a non-negative remainder, R. Then replace (v) by (Q). The quotient and remainder are defined by $(A)_i = (u).(Q)+R$ where $0 \leq R < |(u)|$. $(A)_i$ denotes the initial contents of A.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \longrightarrow X$		
	Clear Q	
	Divide (A) by (X), placing the quotient in Q and leaving R in A	
	$(Q) \longrightarrow X$	
		$(X) \longrightarrow v$ If v is A, Clear A Add D(X) to (A)

OPERATION CODE: 74

INSTRUCTION: Scale Factor, SFuv

FUNCTION: Replace (A) with D(u) unless u is A. Then left circular shift (A) 36 places and continue shifting until $a_{35} \neq a_{34}$. Replace the righthand 15 bits of (v) with the number of left shifts, k, necessary to return the final contents of A, or $(A)_f$, to the original position. The range of k if u is A is $0 \leq k \leq 71$. if u is MC, MD, or Q, k may be 0 or $37 \leq k \leq 71$. Effectively, the initial contents of A, or $(A)_i$, which may be D(u) or D(Q) after the above replacement, are positioned in A_R (with the sign bit represented by A_{35} and the most significant bit by A_{34}) so that $(A)_f = (A)_i . 2^s$. If $0 \leq k \leq 36$, the Scale Factor $s = -k$; if $37 \leq k \leq 71$, $s = 72 - k$. Note that for $0 < k \leq 36$, this positioning scales $(A)_i$ "down"; for $37 \leq k \leq 71$, $(A)_i$ are scaled "up". If $k = 0$, $(A)_i$ were properly positioned before any shifting operations; if $k = 37$, $(A)_i$ are all ones or zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
$(u) \longrightarrow X$ Omit if u is A		
	Clear A Omit if u is A	
	Add D(X) to A	
	Shift (A) left 36 places and continue shifting until $a_{35} \neq a_{34}$	
	Clear X	
	$k \longrightarrow X$	
		$(X_{14} \ldots X_0) \longrightarrow v$ v = Q or A not permissible

OPERATION CODE: 75

INSTRUCTION: Repeat, RPjnw

FUNCTION: This instruction calls for the next instruction, NIuv, to be executed n times, $0 \leq n \leq 2^{12}-1$, its u and v addresses being modified or not according to the value of j. Normally n executions are made, and the program is continued by the execution of the instruction stored at a fixed address F_1. The instruction usually stored at F_1 is MJjv, calling for a jump to its v address, which according to the workings of the Repeat instruction means that (w) of RPjnw will be taken as the next instruction to be executed.

The u and v addresses of the repeated instruction are advanced according to the value of j as follows.

If j = 0, neither the u nor the v execution address of the repeated instruction is advanced.
j = 1, the v execution address of the repeated instruction is advanced after each execution.
j = 2, the u execution address of the repeated instruction is advanced after each execution.
j = 3, both the u and v execution addresses of the repeated instruction are advanced after each execution.

During a repeat sequence, PAK is used as a counter to record the number of times the instruction is executed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
	$w \longrightarrow X$	
		$(X_{14} \ldots X_0) \rightarrow F_1$
	$(PAK) \longrightarrow X$	
	$jn \longrightarrow PAK$	
	Complement PAK	
	$(X) \longrightarrow PCR$	
	Begin Repeat Sequence: Advance (PAK) and investigate its contents. If n = 0, proceed to Normal Repeat Termination If n ≠ 0, execute NI n times,* advancing u and v addresses according to j	

*See Exceptions, discussed later.

Intermediate Steps following each execution of an instruction whose repeat is possible.

FUNCTION: Test for n executions of repeated instruction. Proceed to Normal Repeat Termination if n executions have occurred. If not, repeat instruction.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Advance PAK by one Test PAK for condition indicating n executions of repeated instruction and proceed accordingly.	

Normal Repeat Termination

FUNCTION: Terminate the repeat sequence when an instruction has been executed n times by taking (F_1) as the next instruction. (See Exceptions)

	$(F_1) \longrightarrow X$ $(X) \longrightarrow PCR$	

Abnormalities:

1. If the fixed address F_1 is not a Jump instruction, the address of the instruction to be executed following the instruction stored in F_1 is determined by the modified complement of j from the Repeat instruction. This is the number that remains stored in PAK at the end of a repeat sequence that has a Normal Termination. If, in 75jnw, j was 0, the address will be 40000; if j was 1, the address will be 70000; if j was 2, the address will be 60000; if j was 3, the address will be 50000.

2. In addition to values j = 0, 1, 2, and 3, the value j in 75jnw may also be 4, 5, 6, or 7. An instruction following a Repeat instruction where j = 4, 5, 6, or 7 will be repeated indefinitely unless its execution produces a jump or stop operation or a jump termination. If these latter conditions do not occur the u and v execution addresses of the repeated instruction will be advanced depending on j; if j = 4, neither address is advanced; if j = 5, the v address is advanced; if j = 6, the u address is advanced; if j = 7, both addresses are advanced. If u or v are MD addresses, they may be advanced from 40000 through 77777, and the next advance signal after 77777 will start the address sequence over from 40000. If u or v are MC, Q, or A addresses, they may be advanced 4095, 511 and 511 addresses from their respective first addresses, and the next advance signal will start their address sequence over from their respective first address.

3. If a Repeat instruction is immediately followed by a second Repeat instruction, the second will supersede the first, and the address of the repeated instruction is determined by the number stored in PAK. This address is the complement of "jn-1" which remains stored in PAK from the first Repeat instruction.

TERMINATION OF A REPEAT SEQUENCE

The first step of the Repeat Sequence is to determine whether the value of n in the Repeat instruction is zero or not. If n is zero (indicating that no execution of the instruction following the Repeat instruction is to be made), the attempted Repeat Sequence is immediately terminated according to the Normal Repeat Termination. If $n \neq 0$, the instruction, regardless of what it is, is executed. The Repeat Sequence is concluded by a Normal Repeat Termination, after n executions, except when the instruction following RPjnw is one of the following.

Exceptions:

(1) If the Interpret (14--), Return Jump (37uv), Q Jump (44uv), Sign Jump (46uv), Zero Jump (47uv), Manually Selected Stop (56jv), or Program Stop (57--) instruction is to be executed after a Repeat instruction, the attempted Repeat Sequence is automatically terminated. These instructions are thus executed as if no Repeat instruction preceded them and terminated by the normal Instruction Reference Events.

(2) If either the Index Jump (41uv) or Manually Selected Jump (45jv) is to be executed following a Repeat instruction, they may be repeated n times as specified if no jump operation is called for in the course of their execution. If a jump operation is called for, the instruction is terminated according to the normal Instruction Reference Events, and the content of the jump address is taken for the NI. (No count of the number of times the instruction was executed is retained, however.) If no jump operation is called for in n executions of these instructions, a Normal Repeat Termination is executed and (F_1) is taken for the NI.

(3) If either the Threshold Jump (42uv) or the Equality Jump (43uv) is to be executed following a Repeat instruction, they can be repeated or not depending on whether or not in their execution the threshold or equality conditions are reached and a jump operation is called for. If a jump operation is called for before or exactly after n executions, the Repeat Sequence of these instructions is terminated by a special Jump Termination (see the following page. Note that the Jump Termination is followed by the normal Instruction Reference Events.) The Jump Termination stores the quantity j, n-r in Q and takes the contents of the jump address for the NI. The (Q) can then be used to determine how many times the instruction was executed. If n executions of these instructions are performed and the threshold or equality conditions requiring a jump do not occur, the repeat sequence of these instructions is terminated by the Normal Repeat Termination and (F_1) is taken for the NI.

(4) If the "interrupt" is selected during the execution of a Repeat instruction or an instruction being repeated, the Interrupt Termination sequence concludes the execution of those instructions in paragraphs (1), (2) and (3) above which are stated to be concluded by the normal Instruction Reference Events. When the content of F_1 is taken as the next instruction, the Interrupt Termination sequence concludes the execution of the instruction at F_1.

The Interrupt Termination sequence referred to is found on the page listing the normal Instruction Reference Events.

Jump Termination for repeated instructions 42uv and 43uv.

FUNCTION: When a jump condition, as specified by one of the instructions above, is reached before or immediately after n executions, the steps listed below, followed by the normal Instruction Reference Events, terminate the execution of the instruction.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	(PAK)$\rightarrow$X (X)$\rightarrow$Q v$\rightarrow$PAK	

OPERATION CODE: 76

INSTRUCTION: External Read, ERjv

FUNCTION: If j = 0, replace (v_7 ... v_0) with (IOA) and (v_{35} ... v_8) with zeros; if j = 1, replace (v) with (IOB). (This instruction must be preceded by an External Function instructing the equipment to transmit information to IOB.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X If IOA or IOB has received information from external equipment (IOA) or (IOB) $\longrightarrow$ X Clear IOA or IOB	 (X) $\longrightarrow$ v If v is A, Clear A Add D(X) to A

If IOA is involved, note that when v is a 36-bit register, the eight right-most bits of v are (IOA) and the left-most 28 bits are zeros; when v is A, the eight right-most bits of A are (IOA) and the remaining 64 bits are zeros.

OPERATION CODE: **77**

INSTRUCTION: External Write, EWjv

FUNCTION: If $j = 0$, replace (IOA) with the right-hand eight bits of (v); if $j = 1$, replace (IOB) with (v). Notify external equipment of this transmission to IOA or IOB. (This instruction must be preceded by an External Function instructing the equipment to sense the information from IOA or IOB.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	$(v) \longrightarrow X$ If previous operations involving (IOA) or (IOB) are completed - $(X) \longrightarrow$ IOA or IOB Transmit (IOA) or (IOB) to external equipment	

OPERATION CODE: 44
46

INSTRUCTION: Q-Jump, QJuv
Sign Jump, SJuv

NON-REPEATED

u or v	
MC	9
A	SCC
Q	-

REPEATED*

u or v	
MC	3
A	SCC
Q	-

OPERATION CODE: 45

INSTRUCTION: Manual Jump, MJjv

NON-REPEATED

v	MC	A	Q
	9	SCC	-

JUMP

v	irrelevant
	9

NO JUMP

REPEATED*

v	MC	A	Q
	3	SCC	-

JUMP*

v	irrelevant
	5N

NO JUMP

*See preliminary discussion

INSTRUCTION EXECUTION TIMES

On the following pages the instructions are presented with accompanying tables which give the time required for the execution of each instruction, the execution being repeated or not repeated. The time required for the execution of an instruction is dependent upon the storage references, where such are made, by the u and/or v addresses. The tables for the instructions, with such references applicable and practical, list individually the execution times of each instruction with its u and/or v addresses referencing an MC register, the Accumulator, or the Q Register. (All execution times assume that the instructions are stored in MC.) The specified times are measured, unless otherwise stated, from MPO to MPO. Each number in a table is the number of clock pulse periods (two microseconds each) required for the execution of a particular coded instruction.

Included in the determination of the computer time for an instruction repeated n times are the times required by the Repeat instruction and, in most cases, the instruction stored at F_1. This total time is given by the sum, $27+R_n+p$, where R_n is the execution time of the instruction repeated n times and p is the execution time of the instruction stored at F_1. If n is zero, R_n is zero and, in all cases, the total computer time is 27 + p and the next instruction is taken from F_1.

When certain of the instructions are repeated, the total computer time does not include the time p. These instructions are Interpret 14--; Return Jump, 37uv; Q-Jump, 44uv; Sign Jump, 46uv; Zero Jump, 47uv: Manually Selective Stop, 56uv; and Program Stop, 57--. These instructions are executed only once at the most regardless of an n 1. The next instructions to be executed will not be taken from F_1.

Other exceptions to including the time p in the total computer time are made under certain conditions when the following instructions are repeated: Index Jump, 41uv; Manually Selective Jump, 45jv; Threshold Jump, 42uv; and Equality Jump, 43uv. The time p is not added when, during the repeated n executions of these instructions, the jump requirement is met before or during the nth execution of the instruction. The occurrence of the jump eliminates the procedure of taking the next instruction to be executed from F_1.

Execution times for repeating certain of the instructions are not given if a repeat of such instructions has no practical use.

The following notations are used in the tables of execution times.

SCC	Storage Class Control indicates the coded u and/or v addresses being used are not permissible.
-	Execution time not given for a possible instruction at a Q address.

N	Number of executions of the repeated instruction.
(u_i)	Denotes bit in stage i, $(35 \geq i \geq 0)$, of register addressed as u.
$(u_i)_j$	Contents of u_i during repeat j, j = 1 ... N.
ms	Milliseconds.

OPERATION CODE: 11

INSTRUCTION: Transmit Positive, TPuv

NON-REPEATED

u ╲ v	MC	A	Q
MC	19	17	15
A	15	14	12
Q	16	15	13

REPEATED

u ╲ v	MC
MC	12N+1
A	8N +1
Q	9N +1

OPERATION CODE: 12

INSTRUCTION: Transmit Magnitude, TMuv

NON-REPEATED

u ╲ v	MC	A	Q
MC	19	18	16
A	16	15	13
Q	17	16	14

REPEATED

u ╲ v	MC
MC	12N+1
A	9N+1
Q	10N+1

OPERATION CODE: 13

INSTRUCTION: Transmit Negative, TNuv

NON-REPEATED

u \ v	MC	A	Q
MC	19	17	15
A	15	14	12
Q	16	15	13

REPEATED

u \ v	MC
MC	12N+1
A	8N+1
Q	9N+1

OPERATION CODE: 14

INSTRUCTION: Interpret, IP--

NON-REPEATED

u and v irrelevant
15

OPERATION CODE: 15
16

INSTRUCTION: Transmit U Address, TUuv
Transmit V Address, TVuv

NON-REPEATED

u \ v	MC	A	Q
MC	19	SCC	SCC
A	15	SCC	SCC
Q	16	SCC	SCC

REPEATED

u \ v	MC	A	Q
MC	12N+1	SCC	SCC
A	8N+1		
Q	9N+1		

OPERATION CODE: 17

INSTRUCTION: External Function, EF-v

NON-REPEATED

v	MC	A	Q
	14	11	12

computer operating time, not including a lockout time if IOB is currently in use for output operations.

OPERATION CODE: 21

INSTRUCTION: Replace Add, RAuv

NON-REPEATED

u \ v	MC	A	Q
MC	30	27	28
A	23	20	21
Q	25	22	23

REPEATED

u \ v	MC	A	Q
MC	23N+1	20N+1	21N+1
A	18N		
Q	19N		

OPERATION CODE: 22

INSTRUCTION: Left Transmit, LTjkv

NON-REPEATED

v	MC	A	Q
	16+k	15+k	13+k

REPEATED

v	MC
	(9+k)N+1

OPERATION CODE: 23

INSTRUCTION: Replace Subtract, RSuv

NON-REPEATED

u \ v	MC	A	Q
MC	31	28	29
A	24	21	22
Q	26	23	24

REPEATED

u \ v	MC	A	Q
MC	24N+1	21N+1	22N+1
A	19N		
Q	20N		

OPERATION CODE: 27

INSTRUCTION: Controlled Complement, CCuv

NON-REPEATED

u \ v	MC	A	Q
MC	26	23	24
A	19	16	17
Q	21	18	19

REPEATED

u \ v	MC	A	Q
MC	19N+1	16N+1	17N+1
A	14N		
Q	15N		

OPERATION CODE: 31
32

INSTRUCTION: Split Positive Entry, SPuk
Split Add, SAuk

NON-REPEATED

u	
MC	16 + k
A	13 + k
Q	14 + k

REPEATED

u	
MC	(10 + k) N
	where k is not altered by the Repeat Sequence. For k=0 and 1 use value of k=2.

OPERATION CODE: 33
34

INSTRUCTION: Split Negative Entry, SNuk
Split Subtract, SSuk

NON-REPEATED

u	
MC	17 + k
A	14 + k
Q	15 + k

REPEATED

u	
MC	(11 + k) N
	where k is not altered by the Repeat Sequence. For k=0 and 1, use value of k=2.

OPERATION CODE: 35

INSTRUCTION: Add and Transmit, ATuv

NON-REPEATED

u \ v	MC	A	Q
MC	22	18	19
A	19	15	16
Q	20	16	17

REPEATED

u \ v	MC	A	Q
MC	15N+1	13N	13N
A	12N+1		
Q	13N+1		

OPERATION CODE: 36

INSTRUCTION: Subtract and Transmit, STuv

NON-REPEAT

u \ v	MC	A	Q
MC	23	19	20
A	20	16	17
Q	21	17	18

REPEATED

u \ v	MC	A	Q
MC	16N+1	14N	14N
A	13N+1		
Q	14N+1		

OPERATION CODE: 37

INSTRUCTION: Return Jump, RJuv

NON-REPEATED

u \ v	MC	A	Q
MC	18	SCC	--
A	SCC	SCC	SCC
Q	SCC	SCC	SCC

REPEATED*

u \ v	MC	A	Q
MC	12	SCC	--
A	SCC	SCC	SCC
Q	SCC	SCC	SCC

*See preliminary discussion

OPERATION CODE: 41

INSTRUCTION: Index Jump, IJuv

NON-REPEATED

u \ v	MC	A	Q
MC	27	SCC	-
A	20	SCC	-
Q	22	SCC	-

REPEATED*

u \ v	MC	A	Q
MC	18r+3	SCC	-
A	14	SCC	-
Q	16	SCC	-

where r = number of executions up to the occurrence of the jump.

JUMP*

NON-REPEATED

u \ v	irrelevant
MC	22
A	19
Q	20

REPEATED

u \ v	irrelevant
MC	18N
A	15N
Q	16N

NO JUMP

*See preliminary discussion

OPERATION CODE: 42

INSTRUCTION: Threshold Jump, TJuv

NON-REPEATED

u \ v	MC	A	Q
MC	21	SCC	-
A	18	SCC	-
Q	19	SCC	-

REPEATED*

u \ v	MC	A	Q
MC	15r+5	SCC	-
A	17	SCC	-
Q	18	SCC	-

where r = number of executions up to the occurrence of the jump. The jump will occur during the first execution if u is A or Q.

JUMP*

NON-REPEATED

u \ v	irrelevant
MC	21
A	18
Q	19

REPEATED

u \ v	irrelevant
MC	15N
A	12N
Q	13N

NO JUMP

*See preliminary discussion

OPERATION CODE: 43

INSTRUCTION: Equality Jump, EJuv

NON-REPEATED

u \ v	MC	A	Q
MC	27	SCC	-
A	24	SCC	-
Q	25	SCC	-

REPEATED*

u \ v	MC	A	Q
MC	21r+5	SCC	-
A	23	SCC	-
Q	24	SCC	-

where r = number of executions up to the occurrence of the jump. The jump will occur during the first execution if u is A or Q.

JUMP*

NON-REPEATED

u \ v	irrelevant
MC	27
A	24
Q	25

REPEATED

u \ v	irrelevant
MC	21N
A	18N
Q	19N

NO JUMP

*See preliminary discussion

OPERATION CODE: 47

INSTRUCTION: Zero Jump, ZJuv

NON-REPEATED

u or v	
MC	15
A	SCC
Q	-

REPEATED*

u or v	
MC	9
A	SCC
Q	-

OPERATION CODE: 51

INSTRUCTION: Q-Controlled Transmit, QTuv

NON-REPEATED

u \ v	MC	A	Q
MC	22	18	19
A	19	15	16
Q	20	16	17

REPEATED

u \ v	MC	A	Q
MC	15N+1	13N	13N
A	12N+1		
Q	13N+1		

*See preliminary discussion

OPERATION CODE: 52

INSTRUCTION: Q-Controlled Add, QAuv

NON-REPEATED

u \ v	MC	A	Q
MC	23	19	20
A	20	16	17
Q	21	17	18

REPEATED

u \ v	MC	A	Q
MC	16N+1	14N	14N
A	13N+1		
Q	14N+1		

OPERATION CODE: 53

INSTRUCTION: Q-Controlled Substitute, QSuv

NON-REPEATED

u \ v	MC	A	Q
MC	37	30	32
A	34	27	29
Q	35	28	30

REPEATED

u \ v	MC	A	Q
MC	30N+1	25N	26N
A	27N +1		
Q	28N +1		

OPERATION CODE: 54

INSTRUCTION: Left Shift in A, LAuk

NON-REPEATED

u \ v*	MC	A	Q
MC	22 + k	18 + k	19 + k
A	16 + k	16 + k	SCC
Q	18 + k	SCC	18 + k

REPEATED

u \ v*	MC
MC	(15 + k) N+1

where k is not altered by the Repeat Sequence.

*v is coded with bits $v_6 \ldots v_0$ representing k and bits $v_{14} \ldots v_7$ completing an MC, A, or Q address. If v is an A address, (u) shifted remain in A; if v is a Q address, the final (A_R) are transmitted to Q.

OPERATION CODE: 55

INSTRUCTION: Left Shift in Q, LQuk

NON-REPEATED

u \ v*	MC	A	Q
MC	21 + k	20 + k	18 + k
A	18 + k	18 + k	SCC
Q	17 + k	SCC	17 + k

REPEATED

u \ v*	MC
MC	(14 + k)N+1

where k is not altered by the Repeat Sequence.

*where v is coded with bits $v_6 \ldots v_0$ representing k and bits $v_{14} \ldots v_7$ completing an MC, A, or Q address. If v is an A address, (u) shifted are transmitted to A; if v is a Q address, (u) shifted remain in Q.

OPERATION CODE: 56

INSTRUCTION: Manually Selective Stop, MSjv

NON-REPEATED

v	MC	A	Q
	9	SCC	-

REPEATED*

v	MC	A	Q
	3	SCC	-

NO STOP

v	irrelevant
	2

v	irrelevant
	-5

STOP

OPERATION CODE: 57

INSTRUCTION: Program Stop, PS--

NON-REPEATED	REPEATED*
u and v irrelevant	u and v irrelevant
1	-6

*See preliminary discussion

OPERATION CODE: 61
63

INSTRUCTION: Print, PR-v
Punch, PUjv

NON-REPEATED

v	MC	A	Q
	17	15	16

computer operating time, not including a possible lockout time due to external equipment cycle times of -
Approximately 105 ms for typewriter
16.7 ms for Punch

(cycle times listed are not maximum lockout times)

REPEATED

v	MC	A	Q
	13N	10N	11N

computer operating time, not including lockout times due to external equipment cycle times.

Approximate overall times in ms:
Maximum -
Punch 16.7 (N-1)+12.5
Typewriter 105N
Minimum -
Punch 16.7(N-1)
Typewriter 105 (N-1)

OPERATION CODE: 71

INSTRUCTION: Multiply, MPuv

NON-REPEATED

u \ v	MC	A	Q
MC	58	55	56
A	55	52	53
Q	56	53	54

each plus $2(u_0) + 4\sum_{i=1}^{35} (u_i) + 7\ (u_{35})$

REPEATED

Subtract 36 from corresponding execution times of Multiply Add instruction.

OPERATION CODE: 72

INSTRUCTION: Multiply Add, MAuv

NON-REPEATED

Add 36 to corresponding execution times of Multiply instruction.

REPEATED

u \ v	MC	A	Q
MC	88N	85N	86N
A	85N	82N	83N
Q	86N	83N	84N

Minimum times where $(u_{35})_j$ and $(u_0)_j$ are zeroes and the additional time required for $(u_{34} \ldots u_1)_j$ of ones is given by the summation below the table.

each plus $4 \sum_{j=1}^{N} \left(\sum_{i=1}^{34} u_i \right)_j$

u \ v	MC	A	Q
MC	237N	234N	233N
A	234N	231N	232N
Q	235N	232N	233N

Maximum times where $(u_{35} \ldots u_0)_j$ are all ones.

If the contents of $(u_{35} \ldots u_0)_j$ are known:

u = MC, v = MC

$$88\,N + \sum_{j=1}^{N} \left\{ 2(u_0)_j + 7(u_{35})_j + 4 \sum_{i=1}^{35} (u_i)_j \right\}$$

u = Q, v = Q

$$84N + N \left\{ 2(q_0) + 7(q_{35}) + 4 \sum_{i=1}^{35} (q_i) \right\}$$

OPERATION CODE: 73

INSTRUCTION: Divide, DVuv

NON-REPEATED

u \ v	MC	A	Q
MC	241	240	238
A	238	237	235
Q	239	238	236

add to each $4(a_{71})$

REPEATED

u \ v	MC
MC	$234\ N + 4 \sum_{j=1}^{N} (a_{71})_j + 1$

Maximum time (in case of a preliminary negative remainder)

NON-REPEATED

u \ v	MC	A	Q
MC	238	237	235
A	235	234	232
Q	236	235	233

add to each $4(a_{71})$

REPEATED

u \ v	MC
MC	$231N + 4 \sum_{j=1}^{N} (a_{71})_j + 1$

Minimum time

OPERATION CODE: 74

INSTRUCTION: Scale Factor, SFuv

NON-REPEATED

u \ v	MC
MC	$61 + \gamma$
A	$58 + \gamma$
Q	$59 + \gamma$

REPEATED

if v=A or Q, SCC fault

u \ v	MC
MC	$(54 + \gamma)N+1$

where $\gamma = (36-k) \bmod 72$ and k is the scale factor $0 \leq k \leq 71$. For $k = 37$, use value of $k = 38$.

OPERATION CODE: 75

INSTRUCTION

NON-REPEATED*

$27 + R_n + p$

where p is the execution time of the terminal jump at F_1 and R_n is the execution time of the repeated instruction. (If $n = 0$, $R_n = 0$)

REPEATED

If the RP instruction is repeated, the second one takes precedence over the first.

*See preliminary discussion

OPERATION CODE: 76

INSTRUCTION: External Read, ERjv

NON-REPEATED

v	MC	A	Q
	15	14	12

computer operating time, not including a lockout time if IOB, or IOA, has not yet received the information being "read".

REPEATED

v	MC
	8N+1

computer operating time, not including lockout times.

OPERATION CODE: 77

INSTRUCTION: External Write EWjv

NON-REPEATED

v	MC	A	Q
	14	11	12

computer operating time, not including a lockout time if IOB, or IOA, is currently in use for output operations.

REPEATED

v	MC	A	Q
	10N	7N	8N

computer operating time, not including lockout times.

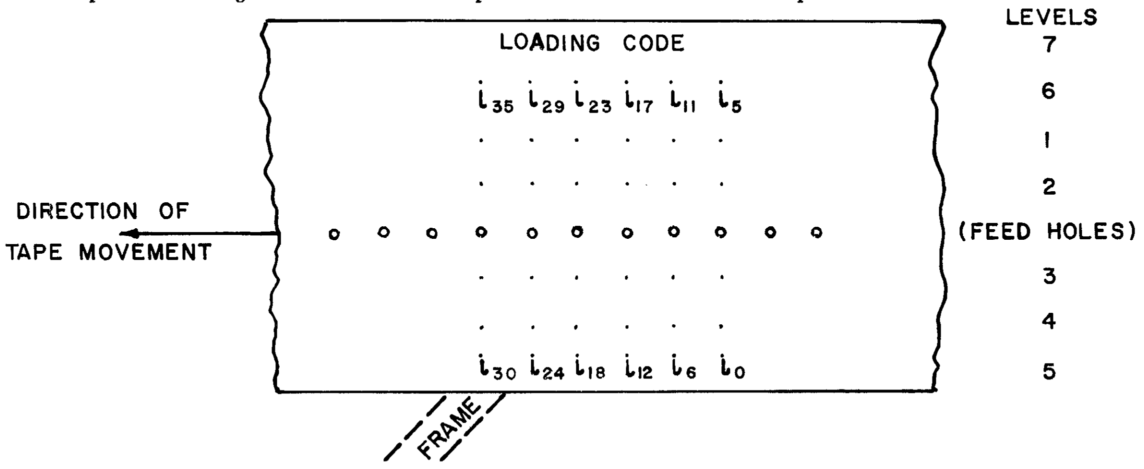
INPUT AND OUTPUT SYSTEMS

1. GENERAL.

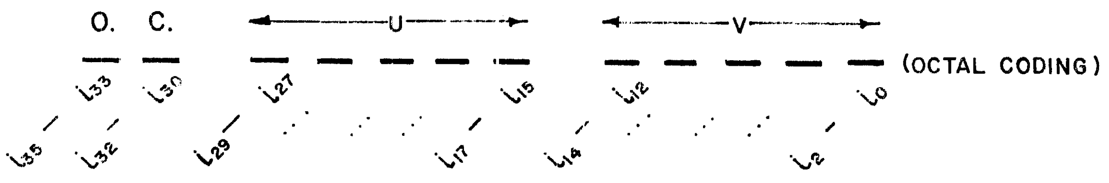
The Input and Output Systems of the Univac Scientific computer provide the means of communication between the computer and the operator. To achieve this communication three things are essential to each input or output system: a medium for external representation of information; external equipment which is capable of (for output) receiving information internally represented and presenting it in its particular external medium of representation, or (for input) sensing external information as it is represented by its particular medium of representation and presenting it to the computer; a means of transmitting and controlling this transfer of information between the external equipment and the computer.

a. STANDARD EQUIPMENT. - The Univac Scientific Computer System includes equipment which utilizes punched paper tape as a means of presenting and receiving information. The Photoelectric Paper Tape Reader senses information punched on paper tape and presents it to the computer; the High Speed Paper Tape Punch receives information from the computer and presents it externally on punched paper tape. The Electric Typewriter is used to create punched paper tape coded to represent typewriter functions and characters. This code is presented to the computer using the tape reader. Typewriter coded information within the computer is presented externally through the typewriter which senses the code and performs accordingly the typewriter function. Typed copy is prepared on standard 8 1/2 by 11 inch typewriter paper and may consist of letters, numbers, signs or symbols, and punctuation marks, spaced in any chosen typewriter format.

Paper tape is described in rows and columns: a single column of positions across the width of a tape is called a frame; frames are divided parallel to the length of the tape into seven levels. Six of these levels are used primarily to represent information to be placed in computer storage; the seventh is used to present loading directions to the computer. A hole punched in any of the six lower levels represents a one; the absence of a hole represents a zero. The tape is thus coded according to binary number notation although the digit grouping may represent decimal numbers, bioctal numbers, or typewriter codes. If the tape is punched to represent a 36-bit operand, $i_{35} \ldots i_0$, the bits representing the word are positioned on the tape as follows:


LEVELS
LOADING CODE
7
i_{35} i_{29} i_{23} i_{17} i_{11} i_5
6
1
2
DIRECTION OF TAPE MOVEMENT
(FEED HOLES)
3
4
i_{30} i_{24} i_{18} i_{12} i_6 i_0
5
FRAME

As shown, six frames are needed to represent a 36-bit word. If the word is a bioctal-coded computer instruction, the digits punched on tape represent the operation code and u and v addresses of the instruction as follows.



b. OPTIONAL EQUIPMENT. - Other input and/or output equipment which is optional with the computer system represents information externally on punched tabulating cards, magnetic tape, or as printed copy.

c. INFORMATION TRANSFER.

(1) INPUT OUTPUT REGISTERS. - Information transmission in and out of the computer is performed as a function of coded computer instructions. It is routed via a buffer (temporary storage) register and the X Register in its transmission between external equipment and an addressed computer location. The buffer registers used are determined by the selection of external equipment for receiving or loading information.

The buffer registers are: the High Speed Punch Register, HPR; the Typewriter Register, TWR; the Input-Output Register A, IOA; the Input-Output Register B, IOB. Since HPR and TWR each provide communication with a specific piece of external equipment, a discussion of these registers is included in the discussion of the proper external equipment. The Input-Output Registers A and B are, respectively, eight stage and 36 stage registers. They are used to provide communication between the computer and the input and/or output equipment for which no specific buffer register has been provided.

(2) EXTERNAL INSTRUCTIONS. - Information transfer to external equipment, via the buffer registers, is initiated by computer instructions which direct the flow of information from an addressed computer location, via the X Register, to the buffer register. The presence of the information in the buffer register allows, under the proper conditions, the external equipment to sense the content of the buffer register and to translate this information and present it externally. Information transfer via the buffer registers from external equipment to the computer is similarly directed by a computer instruction. The information is routed under the proper conditions from the external equipment, via a buffer register and the X Register, to an addressed computer location. The information as represented externally is translated appropriately to its machine representation.

The computer instructions which direct the flow of information are:

PUNCH, PUjv, operation code 63

Addressed computer location $\rightarrow$ X Register $\rightarrow$ HPR $\rightarrow$ Punch

PRINT, PR-v, operation code 61

Addressed computer location $\rightarrow$ X Register $\rightarrow$ TWR $\rightarrow$ Typewriter

EXTERNAL WRITE, EWjv, operation code 77

Addressed computer location $\rightarrow$ X Register $\rightarrow$ IOA or IOB $\rightarrow$ external equipment

EXTERNAL READ, ERjv, operation code 76

External equipment $\rightarrow$ IOA or IOB $\rightarrow$ X Register $\rightarrow$ addressed computer location.

Since no reference to a specific piece of external equipment is made by the External Read and Write instructions which use the IOA and IOB buffer registers for information transfer, a means of selecting the appropriate piece of equipment must also be provided. The External Function instruction, EF-v, uses the IOB register for this purpose. This instruction places ones, as located at address v, in selected stages of IOB and directs IOB Control to then interpret the content of IOB and to accordingly select and initiate a function of external equipment.

EXTERNAL FUNCTION, EF-v, operation code 17

Addressed computer location $\rightarrow$ X Register $\rightarrow$ IOB
Initiate external equipment function according to (IOB)

(3) LOCKOUT CONDITIONS. - During the execution of each of these external equipment instructions, a "lockout" test is made to see if a previous use, if such is the case, of the particular register involved is completed. This is necessary before the current instruction can be completed and before the next instruction can be placed in computer control in readiness for its execution. If the necessary requirements are not met, a lockout condition exists which effectively stops computer operating time.

For output operations a lockout condition exists if the control circuitry of the particular buffer register being referenced by the output instruction has not yet received an indication that the register has been cleared of previous information being routed to external equipment. The condition continues to exist, temporarily stopping computer operations, until the register is cleared, after which the current information may be received by it (machine operations being resumed), and the execution of the instruction completed. The steps involved in an output operation are sequential within the groups as listed below.

Under computer control (for output)

> Content of computer register $\rightarrow$ X register
> TEST FOR LOCKOUT
> (X) $\rightarrow$ buffer register, INITIATE LOCKOUT

Under external equipment control (for output)

> Sense content of buffer register
> Resume signal-Clear buffer register, NULLIFY LOCKOUT INITIATION or CLEAR LOCKOUT IF ESTABLISHED

Computer operations continue after the initiate lockout step unless a second computer output operation involving the same register is begun before the resume signals from external equipment control are received by the computer. If this is the case, the LOCKOUT condition is ESTABLISHED and computer operation is halted at the test lockout step. Computer operation is resumed at its stopping point when the resume signals are received from external equipment.

For input operations a lockout condition exists if the control circuitry of the particular buffer register being referenced by the input instruction has not yet received an indication that the register has received the input information from external equipment. This condition continues to exist, temporarily stopping computer operations, until the information is received, after which computer operations are resumed and the instruction completed. The steps involved in an input operation are sequential within the groups as listed below.

Under external equipment control (for input)

> Input information $\rightarrow$ buffer register
> NULLIFY LOCKOUT INITIATION or CLEAR LOCKOUT IF ESTABLISHED

Under computer control (for input)

> TEST FOR LOCKOUT
> Content of buffer register $\rightarrow$ X
> Clear buffer register, INITIATE LOCKOUT
> Content of X $\rightarrow$ computer storage .

During input operations, a LOCKOUT condition is ESTABLISHED if the computer instruction for input is sensed by computer control before the input information is received by the buffer register referenced by the instruction. If the input information is received by the buffer register before the execution of the input instruction is begun, the lockout initiation condition (left by any previous execution of an input instruction referencing the same register) is nullified, and the input instruction may be executed without lockout delay; but if an input instruction (referencing a register whose previous clearance set up a lockout initiation condition) is attempted before input information is received by the buffer register, a lockout is established. In this case, computer operations are halted at the test lockout step and resumed when the input information is received.

(4) INPUT OUTPUT COMPUTER FAULTS. - Certain conditions may arise during input and output operations which could cause erroneous transmissions to and from external equipment. These conditions do not result in lockouts but cause computer faults which halt computer operation. These faults result from operations involving the Input Output Registers A and B and are indicated on the Supervisory Control Panel, lower left portion of the center section. Since they are classified as Type B computer faults (explained in the section, Operating the Computer), they are also indicated on the control panel in the center section, lower righthand corner, as an IOB fault.

These faults, listed subsequently, are generated because of timing requirements which have not been taken into consideration or because of faulty external equipment.

An IOB (or IOA) Read Fault, Class I, occurs if information is received by IOB (IOA) from external equipment before the control circuitry of the register has received a signal to dispose of information placed there during previous input operations. In other words, if the steps

```
    Under external equipment control (for input)

        Input information ──> buffer register
        NULLIFY LOCKOUT INITIATION or CLEAR LOCKOUT IF ESTABLISHED
```

occur a second time before the steps

```
    Under computer control (for input)

        TEST FOR LOCKOUT
        Content of buffer register ──> X
        Clear buffer register, INITIATE LOCKOUT
        Content of X ──> Computer storage,
```

an IOB (IOA) Read Fault, Class I, is generated.

An IOB (or IOA) Read Fault, Class II, occurs if IOB (or IOA) receives information from external equipment before the control circuitry of the buffer register has received an indication that previous output operations involving it have been completed. In other words, if the steps

```
    Under computer control (for output)

        Content of computer register ──> X register
        TEST FOR LOCKOUT
        (X) ──> buffer register, INITIATE LOCKOUT
```

are followed by the steps

```
    Under external equipment control (for input)

        Input information ──> buffer register
        NULLIFY LOCKOUT INITIATION or CLEAR LOCKOUT IF ESTABLISHED,
```

an IOB (IOA) Read Fault, Class II, is generated.

d. PROGRAM INTERRUPT FEATURE. - The program interrupt feature, discussed previously in the General Description and Sequential Presentation of Instructions sections, permits signals from external equipment to activate the interrupt control and provides automatic interruption of a computer program when the external equipment is ready to communicate with the computer. The instruction at F3, which is acquired for execution by the interrupt feature, could provide a jump (Return Jump) to a subroutine which is coded for input or output operations involving the piece of external equipment which instigated the interrupt. Upon completion of the subroutine, a jump would be programmed, returning operation to the program which was interrupted.

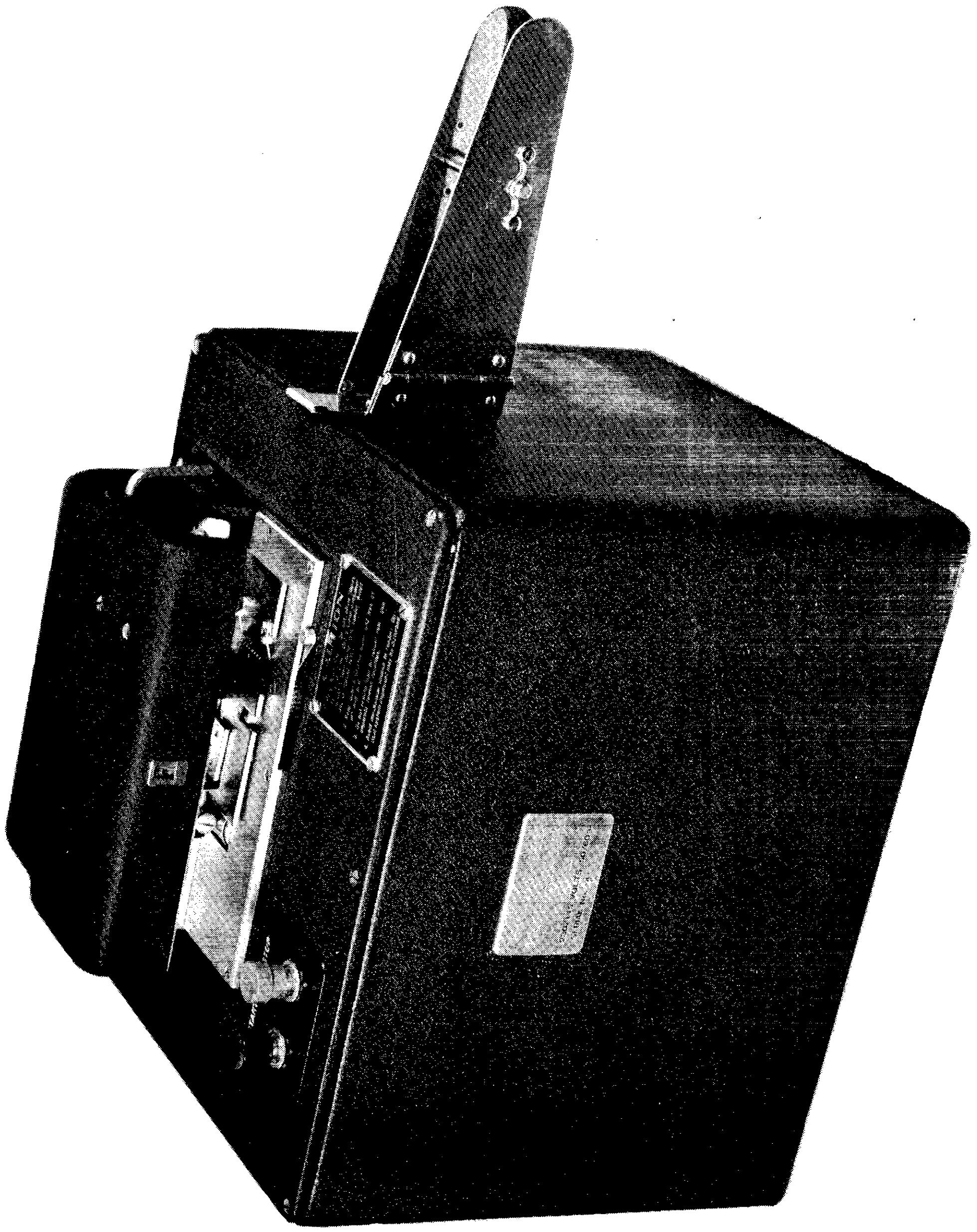
This use of the interrupt allows to a certain extent a disregard of the timing restrictions intrinsic to input and output operations from and to that external equipment which may send an interrupt signal to the computer. Those pieces of external equipment which may thus generate the interrupt signal do so (with the proper conditions) each time the equipment is ready to receive or transmit information. The interruption of the current program must be accomplished, and the input or output subroutine must be executed, within the time allowed by the external equipment for the computer processing of information. Factors could be present in the program being currently executed by the computer which could cause a delay of the interrupt being carried out and consequently delay the necessary input or output operations from being executed within the time limit. Such factors could be an extensive repeat sequence in process at the time the interrupt is activated, or a reference to a drum address which could require up to the maximum access time of 34 milliseconds.

In order for the interrupt signal to be sent to computer (interrupt) control, an interrupt line (where it is provided) must be activated in the piece of external equipment itself. This is done by a switch-setting on the equipment, or by programming an External Function instruction with an IOB Select Interrupt bit.

2. PHOTOELECTRIC PAPER TAPE READER.

a. GENERAL. - The Photoelectric Paper Tape Reader (shown in Figure 1), in conjunction with the IOA and IOB registers and associated control circuitry, provides a means of input of information to the computer. The media of information representation is seven-level punched paper tape. The tape reader has a single reading station composed of a column of seven photocells associated with the seven levels of a tape frame as the tape moves through the reader. (Also punched in the tape are feed holes, parallel to the length of the tape, between levels two and three.)

The six (or seven) bits represented by each frame are sensed simultaneously and transmitted to the six (or seven) lower order stages of IOA. The representation of ones in the tape frame is actually transmitted by a signal received by the corresponding stage of IOA. The representations of zeros are "transmitted" by the absence of such a signal. Thus IOA must initially be clear for an accurate representation of the contents of a frame before such transmissions occur. The transfer of information from IOA to an addressed computer location, via the X Register, is accomplished by programming an External Read instruction, ERjv with j = 0, to be executed each time IOA receives the contents of a frame from the tape reader.

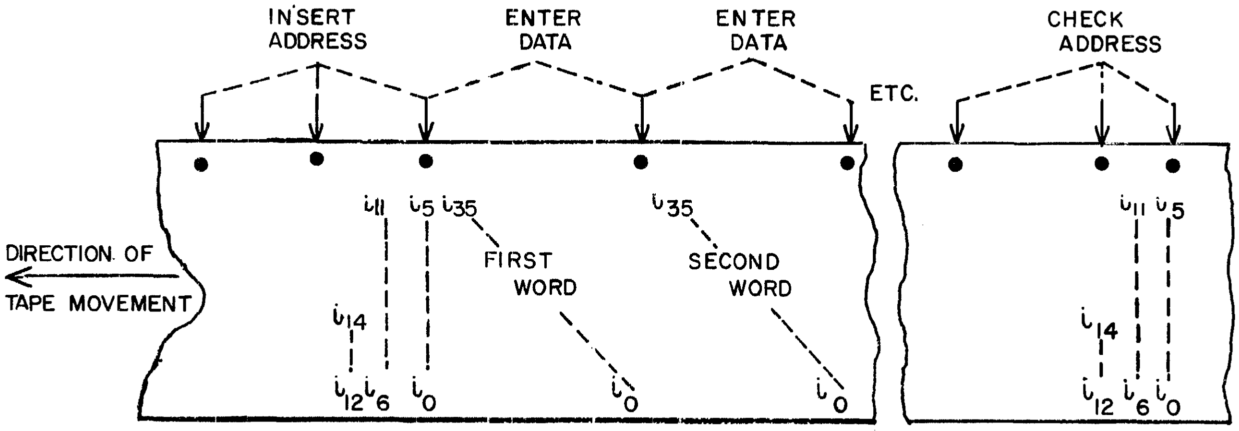


OB 8924

Figure 1. Photoelectric Paper Tape Reader

The six lower levels of the punched tape (6, 1, ..., 5) contain information to be placed at an addressed computer location and may contain loading instructions directing the insertion of information into storage; or the seventh level of the tape may be punched to effect the proper disposal in the computer of the information in the lower six levels. In either case the loading directions in the sixth or seventh levels of consecutive frames must be interpreted within the computer to achieve the designated storage of information.

Typical coding, as punched in the seventh level of a tape, for loading directions is shown on the following diagram. Sprocket or feed holes are not shown on this diagram.



These distributions of holes, as represented by "one's" in the computer, are interpreted as follows.

(1)	INSERT ADDRESS.-	The address designated by $i_{14}...i_0$ to be placed at an addressed location which will serve as a "dummy" PAK.
(2)	ENTER DATA. -	The first word, $i_{35}...i_0$, is to be placed at the address held in the above "dummy" PAK. Successive words are to be placed at consecutive addresses.
(3)	CHECK ADDRESS. -	The address designated by $i_{14}...i_0$ should be identical to the address currently held in the "dummy" PAK.

b. PROGRAMMING FOR INPUT. - Information loading from the tape reader is controlled by using the External Function instruction to place bits in selected stages of IOB. The three stages in which a combination of ones must be placed are, with their corresponding interpretations, as follows.

IOB_{33}	Select Tape Reader
IOB_{16}	Start Tape Reader
IOB_{15}	Stop Tape Reader

The "Select Tape Reader" stage must contain a one for all operations affecting the tape reader. The combinations of one's in IOB which effect control of the tape reader are as follows.

Start (Free Run) - IOB_{33} and IOB16 coded as EF-v,(v) being 10 00002 00000 (octal). This combination causes the reader to start and continue running with the six (or seven) bits of each consecutive tape frame being transmitted to IOA.

Stop - IOB33 and IOB_{15}, coded as EF-v, (v) being 10 00001 00000 (Octal). This combination causes the reader to stop upon the transmission to IOA of the six (or seven) bits of the tape frame sensed by the reader immediately following the interpretation, by the control circuitry of the reader, of (IOB) as a stop.

Step Tape Reader - IOB_{33}, IOB_{16}, and IOB_{15}, coded as EF-v, (v) being 10 00003 00000 (octal). This combination causes the reader to start, transmit to IOA the six (or seven) bits of the first frame of tape it senses, and stop.

The normal rate of tape speed through the reader is approximately 200 frames per second, but, as a function of line voltage, it may be as high as 230 frames per second. Thus the time interval between the sensing of successive frames of tape may vary from 4.3 to 5 milliseconds. Timing factors to be taken into consideration for the three controlled operations of the tape reader are as follows.

(1) Start (Free Run). - Information is being transmitted from the tape to IOA at the rate of one frame every 4.3 milliseconds (assuming for synchronization the shortest time). The contents of IOA should be transmitted to the X Register, using the External Read instruction, at corresponding time intervals for the most efficient use of computer time. The execution of the External Read instruction clears IOA after its contents have been transmitted to X. If another External Read is programmed before the contents of the next frame have been received from the reader, machine operations are temporarily halted by a lockout condition until IOA receives information. If information is transmitted to IOA from the tape reader before IOA has been cleared of the contents of the preceding tape frame, a B Fault occurs and is indicated by the IO Fault light on the Supervisory Control Panel.

(2) Stop. - To insure that the reader is stopped so that the next tape frame whose contents are transmitted to IOA is the last frame desired to be sensed by the reader, the External Function instruction coded for a reader stop should be programmed to be executed at least one millisecond before the sensing of the last desired frame. This can be done by programming it timewise to follow within three milliseconds of the previous External Read. After the reader is stopped, an External Read instruction must be programmed to transmit the contents of IOA to the X Register. If the stop is made during tape loading operations and IOA will not be used before the reader is again started, the External Read may be programmed to precede the start instruction. If the stop instruction is not executed before the transmission to IOA of the contents of the last desired frame, and a read instruction is not programmed to transmit this content of IOA to X, a B Fault will be indicated. The computer will be stopped when IOA is to receive the contents of the next frame which must be sensed before a stop will actually occur.

When a stop occurs, a three millisecond time delay, allowing the reader to stop physically, is generated in the reader control circuitry. This time delay prevents the reader from being started for three milliseconds. Thus, a start instruction, if programmed to be executed during this period, is not effective until the three millisecond period has elapsed.

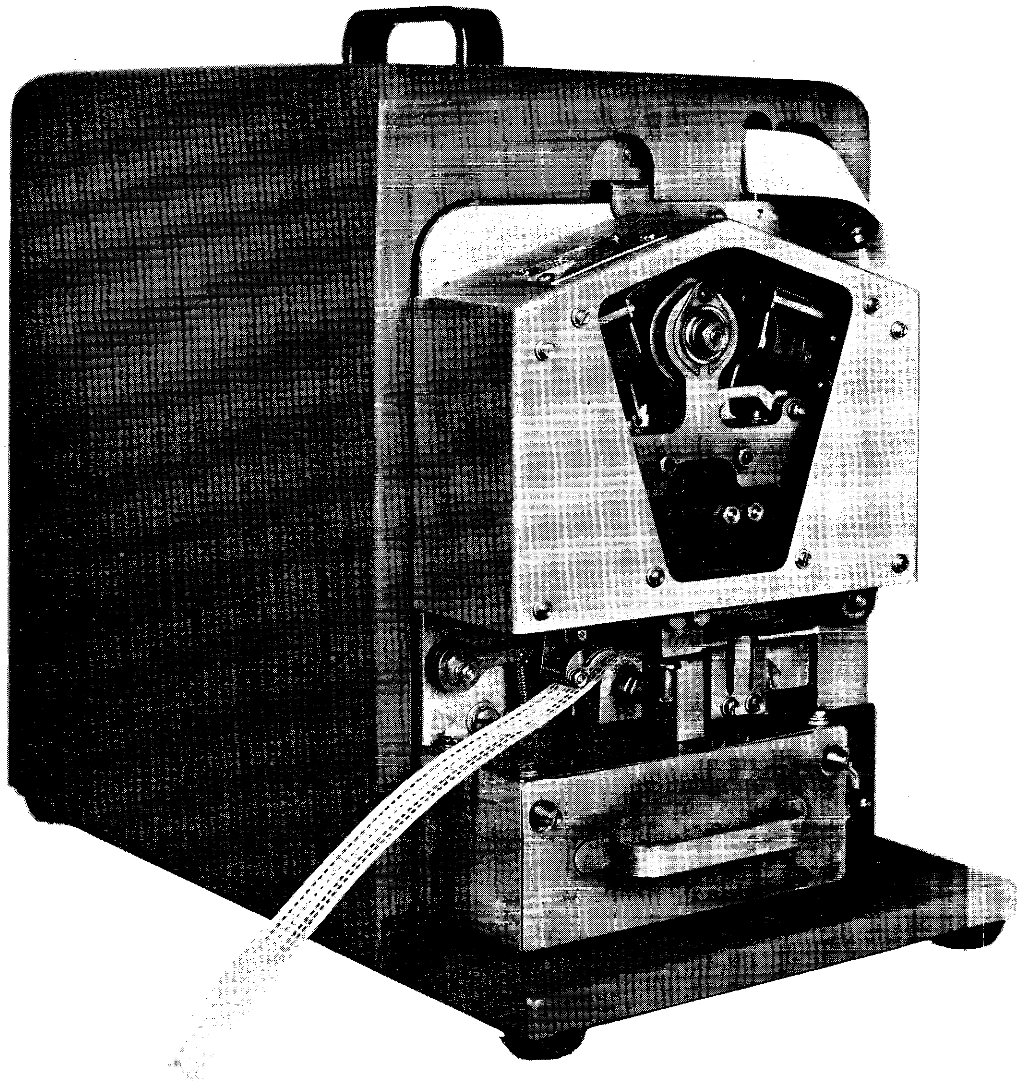
(3) Step. - During step operations the first frame sensed by the reader is the frame which allows the reader stop to be performed. The contents of the first tape frame are transmitted to the IOA register. Thus, consecutively programmed step instructions must each be followed by an External Read instruction with the first preceded by an External Read if IOA is not initially clear. There is a five-millisecond time lapse between successive frames at the operating speed of the reader. A three-millisecond delay is generated by the stop reader operation, and approximately a one-millisecond delay is generated by the reader start operation. Therefore, approximately nine milliseconds are consumed in a step instruction.

c. OPERATION. - Manual operations necessary to the functioning of the Photoelectric Tape Reader consist of inserting the tape to be read in the tape passage of the reader and depressing momentarily the Start button on the reader itself or the Tape Reader Start button on the lower left section of the Supervisory Control Panel of the computer. Depressing either of these buttons turns on the reader power; similarly, depressing either the Stop button on the reader or the control panel turns off the reader power. After the reader power is on, an input routine for the reader may be initiated which includes instructions to start the reader, read the (IOA) to addressed computer locations, and process the information being loaded according to the coded loading directions.

3. HIGH-SPEED PAPER TAPE PUNCH.

The High-Speed (Paper Tape) Punch output system consists of the High-Speed Punch Register, HPR, the High-Speed Punch Control, and the punch (as shown in Figure 2) which presents information on seven-level punched paper tape at the rate of 60 frames per second. The High-Speed Punch Register is a seven-stage buffer register which temporarily stores digits in their transmission between the computer and the punch. The high-speed punch control circuitry senses the contents of HPR and energizes the punch to perforate the tape frame in the levels whose corresponding stages in HPR contain representations of ones. After information is received from HPR, the tape is advanced through the punch by one frame, placing the next tape frame in position to receive information.

The transmission of information to the punch is initiated by the coded computer Punch instruction, PUjv, operation code 63. This instruction directs the transmission, via the X Register, of the contents of the right-hand six stages of v to the first six stages of HPR. Actually, only the representations of ones in stages $X_5 \ldots X_0$ are transmitted to the corresponding stages of HPR. A j of one is transmitted directly to the seventh stage of HPR. According to the contents of $HPR_6 \ldots HPR_0$, holes may be punched in the corresponding tape levels, 7, 6, 1, ..., 5. To transmit from address v a word of 36 bits so that each group of six bits is punched in its correct relative tape position, the Punch instruction must be executed six times, each execution being preceded by a



OA 8787

Figure 2. High Speed Paper Tape Punch

Shift instruction. The word must be shifted at the v address of PUjv six times, the first shift operation positioning the six most significant bits of the word in the six right-most stages of v, and each succeeding shift operation positioning the next most significant six bits in the right-most stages of v.

The High-Speed Paper Tape Punch is ready for operation when the tape is properly positioned in the punch with a "leader" of blank tape preceding the first punching position. This leader may be fed through the punch by depressing either the tape feed button on the computer control panel or on the punch. The punch is started in continuous cycles of operation by setting <u>both</u> the toggle switch on the punch and the tape punch toggle switch on the lower left section of the Supervisory Control Panel of the computer to their ON positions. Either switch in its OFF position prohibits punch operation. The time duration of a punch cycle is 16.7 milliseconds. The first punch instruction initiated during a punch cycle n does not make an effective punch reference, (HPR) $\longrightarrow$ Punch, until the beginning of the next cycle and requires the first 12.5 ms of this succeeding cycle, n + 1, for the completion of its punch reference. Thus HPR is not ready to receive additional output information until the last 4.2 ms of the cycle n + 1. If a second punch instruction is initiated before such an indication of readiness is given to HPR, a computer lockout will occur stopping the computer until 4.2 ms before the beginning of cycle n + 2. If the second punch instruction immediately followed, during cycle n, the first punch instruction, the lockout time is 12.5 ms plus the unexpired time of cycle n after the lockout period effected by the second punch begins. Thus, if the first punch instruction was initiated immediately after the beginning of cycle n, a lockout time of approximately 29 ms would occur (12.5 plus almost 16.7). This is the maximum lockout time possible. The punch reference made by the second Punch instruction will not be completed until the first 12.5 ms of cycle n + 2 have elapsed. If Punch instructions are programmed to be executed at 16.7 ms intervals, a computer lockout may be effected by the second instruction, but the execution of successive Punch instructions will be in synchronization with the punch cycles because of the actual stopping of computer operating time.

4. ELECTRIC TYPEWRITER.

The Electric Typewriter output system consists of the Typewriter Register, TWR, the Typewriter Control, and the Electric Typewriter, as shown in Figure 3. The Typewriter produces typewritten manuscript with letters in upper and/or lower case, numbers of the decimal system typed on the same level as letters or elevated above this level as exponents, and a variety of characters such as a plus sign (+), a minus sign (-) , and a comma (,), etc. These letters, numbers, and characters are typed in a format determined by the response of the typewriter control circuitry to six-bit codes which are interpreted as, and actuate the typewriter to, such physical operations as space, carriage return and tabulator. Incidental to a carriage return is a line spacing operation, the number of spaces between consecutive lines being determined by the manually set line spacer on the typewriter. The tabulator operation also requires a manual setting; the interpretation of the tab code as such releases the carriage to move to a predetermined manual tab setting. A shift-up operation positions the type bars for upper case typing. To resume typing in lower case a shift-down operation must be executed.

Letters and numbers are also typed, as directed by the Typewriter Control, in response to six-bit codes with each code representing a single letter or number. To type a letter in upper case or a number as an exponent, the type bars must be shifted to their upper case position before the typewriter responds to the letter or number code. Two characters are represented by a single code. The character which is typed is determined by the location of the type bars in their upper or lower case position.

A list of the typewriter codes, given as two-digit octal numbers, is presented with their associated typewriter functions in Table 1. If an illegal code is sent to the typewriter, an A Fault will occur, which is indicated by the Print fault light on the Supervisory Control Panel. The computer is stopped in its operations at the point when its control receives a signal from the typewriter control that an illegal code has been detected. The computer may be restarted from where it ceased operations by first pushing the CLEAR A FAULT button and then by pushing the START button on the computer. This insures that the illegal code is cleared from TWR.

Typewriter operations for output are initiated by the computer instruction Print, PR-v, operation code 61. This instruction directs the transmission, via the X Register, of the contents of the right-hand six stages of v to the six stages of the Typewriter Register. TWR serves as a buffer register for information being transmitted from the computer to the typewriter. Actually, only the representations of ones in stages $X_5 \ldots X_0$ are transmitted to the corresponding stages of TWR. The typewriter control circuitry senses the representation of a six-bit code in TWR and in so doing causes the typewriter to type a single letter, number, or character or perform a single typewriter operation. For example, to type the single upper case letter M, the following two instructions would be programmed.

61-v, $(v_5 \ldots v_0)$ is 100 111

61-v, $(v_5 \ldots v_0)$ is 000 111

A movement left of the carriage by one space to reposition it for the next typing operation occurs automatically after each typing of a letter, number, or character.

The speed of the typewriter allows it to type approximately nine characters per second. Thus, a timing period of approximately 105 ms will elapse between consecutive prints by the typewriter. (Coded typewriter functions require a longer timing period for their completion.) If successive Print instructions are programmed to be executed in less time, a lockout condition will exist until the control circuitry of TWR receives an indication that the typewriter is ready to receive another six-bit code. The maximum lockout time for the typewriter is dependent upon which phase of the typewriter cycle is being executed currently when a test lockout reference of a successive Print instruction is made.

Manual preparation of the typewriter for operation consists of inserting paper, setting the OFF-ON switch to the right of the typewriter keyboard to the ON position, and selecting the ON LINE switch position.

TABLE 1. TYPEWRITER CODES

The upper case, UC, or lower case, LC, character is typed according to the position of the type bars.

Type Letter UC	Type Letter LC	Octal
A	a	30
B	b	23
C	c	16
D	d	22
E	e	20
F	f	26
G	g	13
H	h	05
I	i	14
J	j	32
K	k	36
L	l	11
M	m	07
N	n	06
O	o	03
P	p	15
Q	q	35
R	r	12
S	s	24
T	t	01
U	u	34
V	v	17
W	w	31
X	x	27
Y	y	25
Z	z	21

Type Number UC	Type Number LC	Octal
1	1	52
2	2	74
3	3	70
4	4	64
5	5	62
6	6	66
7	7	72
8	8	60
9	9	33
0	0	37

Perform Type-writer Operation	Octal
Space	04
Shift up	47
Shift down	57
Back space	61
Car. return	45
Tabulator	51
Color Shift	02
Code delete	77
Stop	43

Type Symbol UC	Type Symbol LC	Octal
- (Superscript Minus)	- (Hyphen or Minus)	56
. (Multiply)	= (Equals)	44
/ (Virgule)	+ (Plus)	54
((Open Parens)	, (Comma)	46
) (Close Parens)	. (Period)	42
_ (Underline)	| (Absolute)	50

5. PUNCHED CARD INPUT/OUTPUT SYSTEM.

a. GENERAL. - The Punched Card Input/Output equipment, in conjunction with the IOA and IOB registers and associated control circuitry,provides a means of input to and output from the Univac Scientific computer. The input/output card equipment comprises the Card Unit, shown in Figure 4, and the Card Unit Control. The media of information representation is 80-column tabulating cards, in which holes are punched in specified patterns. These cards may be punched or read at a rate of 120 cards per minute.

The Card Unit, as used with the Univac Scientific, utilizes a left-hand channel for punching cards and a right-hand channel for reading cards, Punched cards whose information content is to be transmitted to the computer are routed through the "read" channel of the Card Unit; blank cards which are to receive information and to be punched accordingly are routed through the "write" channel. Cards to be routed through the read channel are placed in the card read feed hopper and received, after their advancement through the channel, by the read receiving stacker. Cards to be routed through the write channel are placed in the card punch feed hopper and received, after their advancement through the channel, by the punch receiving stacker.

The tabulating card used by the Card Unit is divided into 12 horizontal rows and 80 vertical columns with groups of these columns being designated as Fields I, II, and III as shown in the diagram of an unpunched card in Figure 5. A rectangular hole, or index, may be punched at the intersection of any row and column. If the input or output information is coded to represent binary words, each row may be punched to represent two 36-bit words plus six additional bits of information. (A hole represents a one; the absence of a hole denotes a zero.) Tabulating cards may also be punched in a pattern to represent alphabetical and digital characters. An optional punched pattern of letters and decimal digits is shown in Figure 6.

b. PROGRAMMING FOR INPUT AND OUTPUT. - Information is transmitted to or from the cards in response to programmed External Function and External Write or External Read instructions. A series of three of these EW or ER instructions, programmed consecutively, direct the transmission of each row of information, via X, between addressed computer locations and the Input-Output Registers, IOA and IOB. The sequence of information flow is as follows.

To transmit information from the computer to a tabulating card, the group of following External Write instructions, preceded by appropriately coded External Function instructions, is executed twelve times, once for each card row.

EWjv, j = 0	(X) $\longrightarrow$ IOA $\longrightarrow$ Field III	
EWjv, j = 1	(X) $\longrightarrow$ IOB $\longrightarrow$ Field I	Row 9, 8, ... 0, 11, 12
EWjv, j = 1	(X) $\longrightarrow$ IOB $\longrightarrow$ Field II	

To transmit information from a punched tabulating card to the computer, the group of following External Read instructions, preceded by appropriately coded External Function instructions, is executed twelve times, once for each card row.

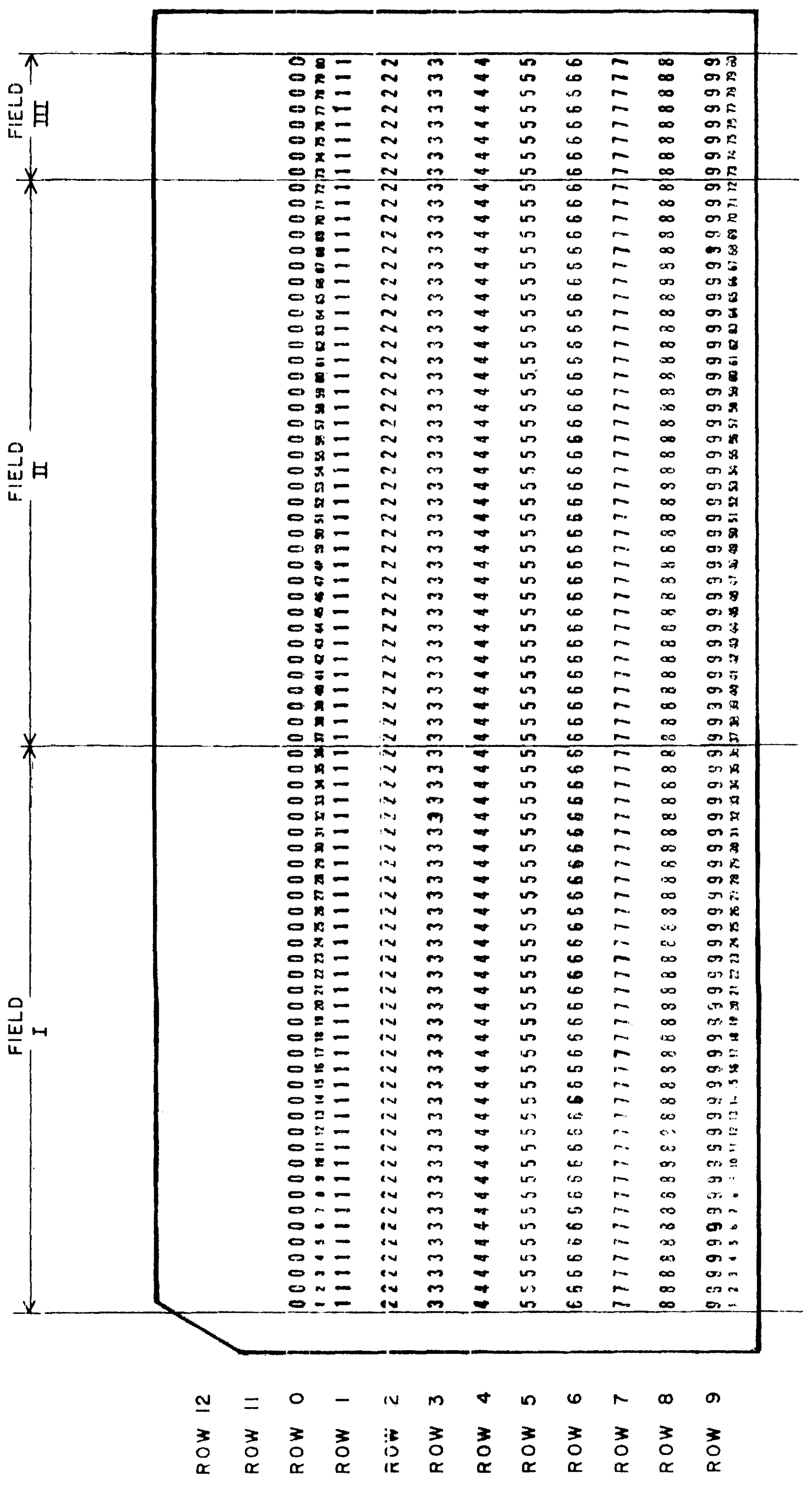


Figure 5. Tabulating Card Fields, Columns, and Rows

PX 38

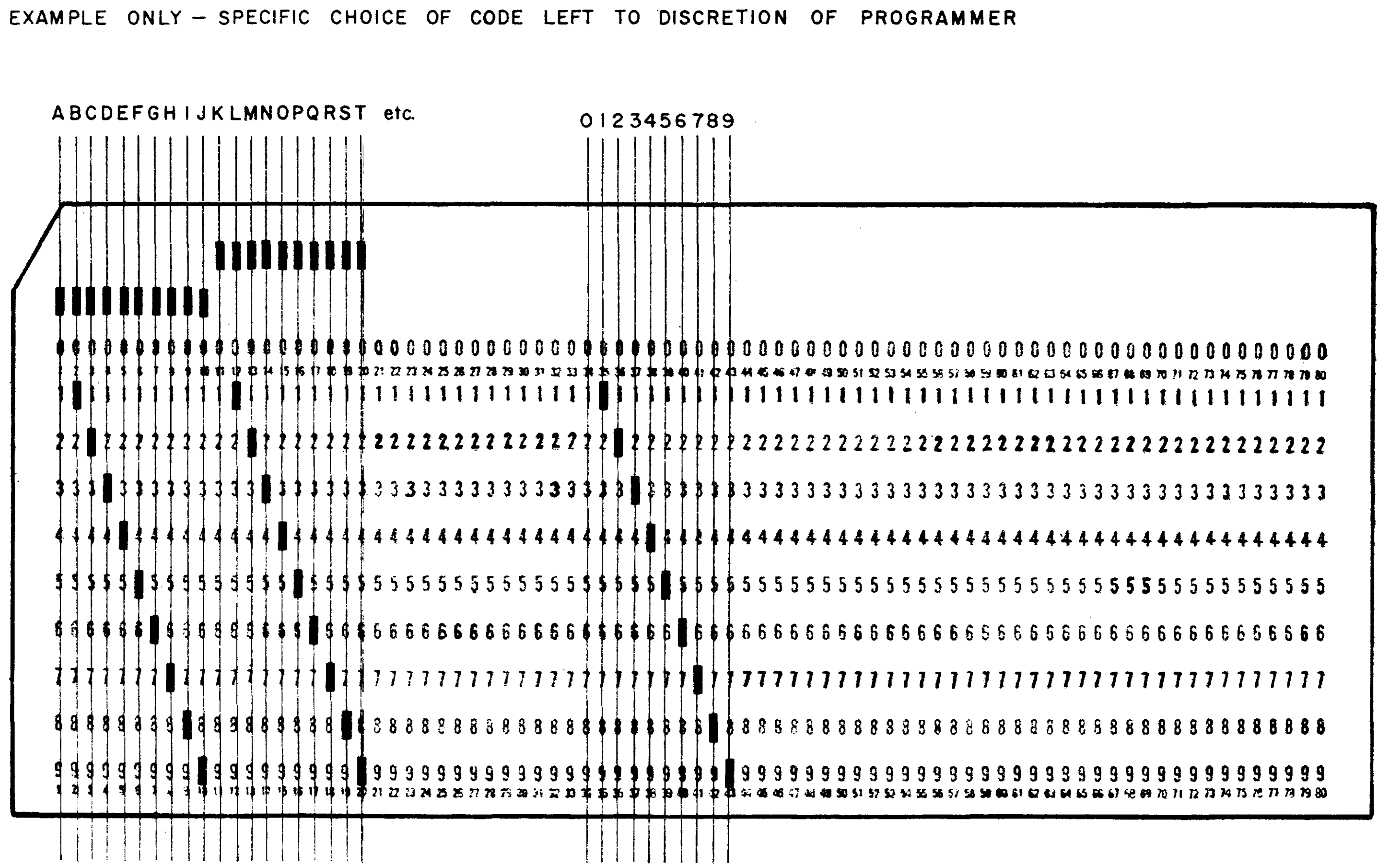

EXAMPLE ONLY — SPECIFIC CHOICE OF CODE LEFT TO DISCRETION OF PROGRAMMER
ABCDEFGHIJKLMNOPQRST etc.
0123456789

Figure 6. An Example of Punched Card Alphabetical and Digital Representation

PX 38

ERjv, $j = 0$		Field III$\rightarrow$(IOA)$\rightarrow$ X
ERjv, $j = 1$	Row 9, 8, ... 0, 11, 12	Field I $\rightarrow$(IOB)$\rightarrow$ X
ERjv, $j = 1$		Field II$\rightarrow$(IOB)$\rightarrow$ X

Information transmission between IOA and Field III is optional and may be omitted by not executing EW or ER with $j = 0$ and by manually setting a switch on the Card Unit Control cabinet to its "Out" position. For reading and writing of all three fields the position of this switch, called the Enable Field III switch, is in its "Normal" setting.

Preceding any reading and writing operations by the punched card system, at least one External Function instruction must be executed to start the Card Unit, position the cards for reading and/or punching (these two conditions may be established also by manual operation), and establish the condition necessary for a read and/or write. The External Function EF-v transmits representations of ones to certain stages of IOB, then, according to the selected stages, the following signals are sensed by the card system.

IOB_0	Read
IOB_1	Punch
IOB_2	Pick Reader Card
IOB_3	Pick Punch Card
IOB_4	Stop (and Drop Selections)
IOB_5	Run Free
IOB_7	Interrupt
IOB_{35}	Start Cycle

Start Cycle - This selection must be present in each EF-v to make any of the other selections effective. A signal from IOB_{35} engages the drive mechanism of the Card Unit and causes one "cycle" of operation. The time consumption of each cycle is 500 milliseconds with each cycle consisting of a sequence of 18 cycle points of approximately 27.8 milliseconds each. The cycle points are numbered in the order 14, 15, 16, 17, 18, 9, 8, ... 1, 0, 11, 12, 13, (14).

Interrupt - With the Interrupt switch on the Card Unit Control cabinet set to "By Command", a selection of IOB_7 energizes an interrupt line in the card equipment so that an interrupt signal is sent to the computer (interrupt) control when the conditions are established for reading and/or writing a card row. This occurs each time a card row is in position for reading and/or writing (i.e., at the beginning of cycle points 9, 8, ..., 1, 0, 11, 12) if the Start (or Run Free) and Read and/or Punch selections are effective. The first External Read instruction must be executed within 10 milliseconds after the interrupt signal is sent to the computer. The first External Write instruction must be executed within 1.5 milliseconds after the interrupt signal is sent to the computer.

The interrupt selection may be made by setting the INTERRUPT switch on the punched card equipment to its "Locked In" position. With this setting the interrupt line in the equipment is energized (without the IOB_7 selection) under the conditions mentioned above.

Run Free - A selection of IOB_5 causes continuous cycle operation by retaining in effect the Start selection, along with other selections made simultaneously with the Run Free, until a Stop signal is received by the execution of another programmed EF-v.

Stop - A selection of IOB_4 permits only one more cycle of operation with the selections effective that were made by the previous EF-v; i.e., the stop is sensed after point zero which allows another complete cycle before the Stop is effected.

Pick Punch Card - A selection of IOB_3 causes the bottom-most card in the punch card feed hopper to be withdrawn from the hopper and placed in the punch channel. This selection is also necessary for the first advancement of the card through the channel.

Pick Reader Card - A selection of IOB_2 causes the bottom-most card in the read card feed hopper to be withdrawn from the hopper and placed in the read channel. This selection is also necessary for the first advancement of the card through the channel.

Punch - A selection of IOB_1 enables the punch mechanism to receive information from IOB (and IOA) and prepares it to punch this information on a card during the next cycle.

Read - A selection of IOB_0 enables the brushes used for reading to sense the information in each row of a card as it passes beneath them.

After these IOB select signals are received by the punch card equipment they are held in effect for one cycle of operation (or a number of cycles if a Run Free Select was chosen) by the control circuitry so that IOB may be cleared for information transmission.

After a card is picked and withdrawn from either the punch card feed hopper or the read card feed hopper, it may advance through a series of five positions or "stations" in the read channel or punch channel. A Pick Reader Card selection or a Pick Punch Card selection places a card in either Read Station 1 or Punch Station 1. A Pick Punch Card Selection or Pick Read Card selection is necessary also to advance a card in the channel from Station 1 to Station 2. These operations may be initiated by programming External Function instructions with bits in the appropriate stages of IOB or by manually operating switches on the Card Unit itself. After a card is in Station 2, in either the punch or read channels, four more cycles of operation are necessary to advance the card through the channel and into its final position in the receiving stacker. Each cycle of operation advances cards in both channels if they are in or past Station 2 to the next station.

Punch Station 3 in the punch channel contains the punch mechanism. After a card is in Punch Station 2, one cycle of operation is needed to advance the card to the punching position. During the cycle which advances the card from Punch Station 2 to Punch Station 3, the information to be punched must be received from IOB (and IOA). More explicitly, during each cycle point 9, 8, ... 1, 0, 11, 12, of 27.8 milliseconds, the series of three (or two) External Write instructions should be executed consecutively, and the Punch select signal must be in effect. With these conditions information to be punched (in the correspondingly numbered card rows 9, 8, ..., 0, 11, 12) can be transmitted to the punch mechanism. At the beginning of the next cycle, automatic punching of a complete card occurs before the card is advanced to Station 4. Outlines of programming for punching of consecutive cards are given in Tables 2 and 3.

Read Station 2 in the read channel contains the brushes used for sensing the cards. During the cycle of operation in which the card is advanced through the read station, the information may be read, a row at a time, as the card is advanced past the read brushes. This information must be transmitted to computer storage from IOB (and IOA) as it is received. More explicitly, during each point 9, 8, ... 1, 0, 11, 12 of 27.8 milliseconds, the series of three (or two) External Read instructions should be executed consecutively, and the Read select signal must be in effect. This enables the transmission to addressed storage locations of the information of the correspondingly numbered rows 9, 8, ... 1, 0, 11, 12. Outlines of programming for reading of consecutive cards are given in Tables 4 and 5.

An outline of programming for simultaneous reading and writing of consecutive cards is given in Table 6 and on the page preceding it. The transmission of Field III in any of the programs may be eliminated by not executing the External Read or Write instructions with j = 0 and by setting the Enable Field III switch properly.

c. OPERATION.

(1) COMPUTER CONTROLLED. - The equipment is prepared for controlled operation, after power is applied at the Card Unit Control Cabinet, by following the steps below.

Step 1 - If only 72-card columns are to be read or punched, set the FIELD III switch (S01) to the "Out" position. If 80 columns are to be read or punched, set this switch to the "Normal" position.

Step 2 - If card reading is to be performed, place the deck of cards to be read into the right-hand feed hopper of the Card Unit. If card punching is to be performed, place a deck of cards in the left-hand feed hopper. Place the metal weights on top of the decks. In either hopper, cards are placed face (printed side) down, so that the "9" edge enters the channel first.

Step 3 - Set the toggle switches on the Card unit in the following manner.

DUPL.	away from operator
PUNCH	away from operator
MOTOR	left
DC	left
READ	away from operator
PICK READ	towards operator
PICK PUNCH	towards operator
STANDBY	towards operator

After these steps have been performed, the punched card program may be started in the computer.

TABLE 2. WRITE - SINGLE OR CONSECUTIVE CARDS

The computer instructions below withdraw two cards from the punch card feed hopper, position the first card for punching and punch information in it, and continue advancing it through the punch channel until it reaches its final position in the receiving stacker. (The second card withdrawn from the hopper is left in the first station.) Other instructions may be programmed and executed during and between each cycle of operation of the card equipment providing that the timing requirements noted are met.

EF-v	(v) = Start Pick Punch Card	1 cycle
EF-v	(v) = Start Pick Punch Card	1 cycle
EF-v EW,0,v EW,1,v EW,1,v	(v) = Start Punch Within 140.5 ms of the start of this cycle the execution of the following three instructions should be initiated: } Repeat for each card row, each repetition being initiated not later than 1.5 ms after the beginning of the corresponding point.	1 cycle
EF-v	(v) = Start (punching occurs)	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start (channel cleared of last punched card)	1 cycle

To transmit information to n consecutive cards, without selecting the Run Free bit, include a Pick Punch Card bit in (v) of the third External Function instruction above. Repeat the third EF-v instruction and the group of External Writes occurring in that cycle n times. Then, at the conclusion of the program above, the nth or last punched card will be found in the punch stacker with cards remaining in the first and fifth punch stations.

TABLE 3. WRITE - FREE RUN

The computer instructions below withdraw singly a sufficient number of cards from the punch card feed hopper to hold the information to be punched, position them for punching and punch information in them, and continue advancing them through the punch channel until the last card punched reaches its final position in the receiving stacker. (Unpunched cards are left in the first and fifth stations.) Other instructions may be programmed and executed during and between each cycle of operation of the card equipment providing the timing requirements are met.

Number n is the number of cards required to hold the information to be written.

EF-v	(v) = Start Pick Punch Card	1 cycle
EF-v	(v) = Start Pick Punch Card	1 cycle
EF-v EW,0,v EW,1,v EW,1,v	(v) = Start Free Run Pick Punch Card Punch Within 140.5 ms of the start of this cycle the execution of the following three instructions should be initiated: } Repeat for each card row, each repetition being initiated not later than 1.5 ms after the beginning of the corresponding point. The repetitive group of External Write instructions must be programmed n-1 times, each group being executed timewise within the cycle which enables the punching.	repeated n-1 cycles
EF-v EW,0,v EW,1,v EW,1,v	(v) = Start Stop (punching of n-1 card occurs) Within 140.5 ms of the start of this cycle the execution of the following three instructions should be initiated. } Repeat for each card row, each repetition being initiated not later than 1.5 ms after the beginning of the corresponding point.	1 cycle
EF-v	(v) = Start (nth card punched)	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start (nth card placed in receiving stacker)	1 cycle

TABLE 4. READ - SINGLE OR CONSECUTIVE CARDS

The computer instructions below withdraw two cards from the read card feed hopper, position the first card for reading, transmit its contents to the computer, and continue advancing it through the read channel until it reaches its final position in the receiving stacker. (The second card withdrawn from the hopper is left in the first station.) Other instructions may be programmed and executed durina and between each cycle of operation of the card equipment providing the timing requirements are met.

EF-v	(v) = Start Pick Read Card	1 cycle
EF-v ER,0,v ER,1,v ER,1,v	(v) = Start Pick Read Card Read Within 149.0 ms of the start of this cycle the execution of the following three instructions should be initiated. } Repeat for each card row, each repetition being initiated not later than 10 ms after the beginning of the corresponding point.	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start (channel cleared of last read card)	1 cycle

To transmit information from n consecutive cards, without selecting the Run Free bit, repeat the second External Function instruction above and the group of External Reads occurring in that cycle n times. Then at the conclusion of the program above, the nth or last read card will be found in the read stacker with an n + 1 card remaining in the first read station.

TABLE 5. READ - FREE RUN

The computer instructions below withdraw singly from the read card feed hopper the cards to be read, position them for reading and transmit their information content to the computer, and continue advancing them through the read channel until the last card read reaches its final position in the receiving stacker. (A card is left in the first station.) Other instructions may be programmed and executed between and during each cycle of operation of the card equipment providing the timing requirements are met.

Number n is the number of cards to be read from.

EF-v	(v) = Start Pick Read Card	1 cycle
EF-v ER,0,v ER,1,v ER,1,v	(v) = Start Free Run Pick Read Card Read Within 149.0 ms of the start of this cycle the execution of the following three instructions should be initiated. } Repeat for each card row, each repetition being initiated not later than 10 ms after the beginning of the corresponding point. The repetitive group of External Read instructions must be programmed n-1 times, each group being executed timewise within the cycle which enables the reading.	repeated n-1 cycles
EF-v ER,0,v ER,1,v ER,1,v	(v) = Start Stop (reading of nth card) Within 149.0 ms of the start of this cycle the execution of the following three instructions should be initiated. } Repeat for each card row, each repetition being initiated not later than 10 ms after the beginning of the corresponding point.	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start (nth card placed in receiving stacker)	1 cycle

SIMULTANEOUS READ AND WRITE - SINGLE OR CONSECUTIVE CARDS

The outlined programs for reading and writing single or consecutive cards simultaneously may be deduced by interposing the two outlined programs for read and write, single or consecutive cards. A composite of the selections made in both programs in each individual cycle would be selected for one cycle of the simultaneous operations. If it is desired that the reading occur during the same cycle that the information to be punched is being received by the card equipment, the second cycle of operation of the program for writing should be interposed with the first cycle of operation of the reading program. External Read instructions immediately follow External Write instructions when both occur during the same cycle.

The same timing considerations must be given the simultaneous performance of reading and writing as when they occur individually. Other instructions may be programmed and executed between and during each cycle of operation providing the timing requirements are met.

TABLE 6. SIMULTANEOUS READ AND PUNCH - FREE RUN

The computer instructions below withdraw cards from the read card feed hopper and punch card feed hopper, position them for reading and writing, perform the information transmittal, and continue advancing them through their individual channels until the last cards read and punched reach their final positions in the receiving stackers. Other instructions may be programmed and executed between and during each cycle of operation of the card equipment providing the timing requirements are met.

Number n is the number of cards to be read from and the number of cards required to hold the information to be written.

EF-v	(v) = Start Pick Punch Card	1 cycle
EF-v	(v) = Start Pick Punch Card Pick Read Card	1 cycle
EF-v EW 0,v EW 1,v EW 1,v ER 0,v ER 1,v ER 1,v	(v) = Start Free Run Pick Read Card Read Pick Punch Card Punch Within 140.5 ms of the start of this cycle the execution of the following six instructions should be initiated. Repeat for each card row, each repetition being initiated not later than 1.5 ms after the beginning of the corresponding point. The repetitive group of External Read and External Write instructions must be programmed n-1 times, each group being executed timewise within the cycle which enables the reading and writing.	repeated n-1 cycles

TABLE 6. SIMULTANEOUS READ AND PUNCH - FREE RUN

EF-v	(v) = Start Stop Within 140.5 ms of the start of this cycle the execution of the following six instructions should be initiated.	
EW ,0,v EW ,1,v EW ,1,v ER ,0,v ER ,1,v ER ,1,v	Repeat for each card row, each repetition being initiated not later than 1.5 ms after the beginning of the corresponding point.	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start	1 cycle
EF-v	(v) = Start (nth card placed in write receiving stacker)	1 cycle
EF-v	(v) = Start (nth card placed in read receiving stacker)	1 cycle

(2) MANUAL OPERATION. - The punched card system may be operated manually by switches and pushbuttons on the panel located below the feed hoppers. The procedures are as follows.

(a) TO START. - If cards are in the feed hoppers and it is desired to drive the card equipment through one or more card cycles, press the START pushbutton.

If the START button is pressed and released immediately, the card equipment will drive through one complete card cycle. If the button is held down, the equipment will run continuously.

(b) TO PICK PUNCH CARDS. - If it is desired to feed cards into the punching channel, set toggle switch PICK PUNCH CARDS to the ON position (away from the operator), then hold down the START pushbutton until the desired number of cards have been inserted.

(c) TO PICK READ CARDS. - If it is desired to feed cards into the reading channel, set toggle switch PICK READ CARDS to the ON position (away from the operator), then hold down the START pushbutton until the desired number of cards have been inserted.

(d) TO STOP. - To stop the card equipment during a controlled run so that cards may be added to the feed hoppers, etc., either set toggle switch STANDBY to the ON position (away from the operator) or press and hold down the red STOP pushbutton. This causes the card equipment to stop at the end of the current card cycle so that the computer program is temporarily halted. If the toggle switch is reset or the red STOP pushbutton is released the system resumes normal operation.

To clear cards from either channel, remove the cards from the hopper, set the PICK READ CARD or PICK PUNCH CARD switch to its ON position if a card is in Station 1, and depress the CLEAR and START buttons until the channel is clear.

(3) FAULTS. - When certain faults occur in the system, the card equipment stops. Certain of these faults result in B Fault computer stops which allow the equipment to complete its current cycle before it stops. A few of these faults are described below.

(a) OVERHEAT. - If the temperature in any portion of the Card Unit Control cabinet rises above 100°F, a computer A Fault stop occurs, and the amber OVERHEAT indicator on the end of the control cabinet glows. The Overheat fault in itself will not cause a card equipment stop, but it may lead to a No Information A Fault which causes a B Fault computer stop and a card equipment stop.

(b) NO INFORMATION. - If the computer program specifies that punching and/or reading is to be executed but insufficient or tardy External instructions are being executed, a B Fault computer stop occurs and the amber NO INFORMATION indicator on the end of the Card Unit Control cabinet glows. The computer program must be restarted from the beginning if this occurs.

(c) VOLTAGE FAULT. - If power to the card equipment drive motor fails, the VOLTAGE FAULT indicator on the end of the Card Unit Control cabinet glows. The fault may be produced by faulty wiring or by blown fuses.

(d) NO CARD IN READER. - If card reading is supposed to occur and no card is present in the reading station, a B Fault computer stop is produced and the NO CARD IN READER indicator on the Card Unit Control cabinet glows. This fault may be caused by failure to feed cards or by errors in the computer program. When the source of error is removed, the computer program must be restarted from the beginning.

(e) NO CARD IN PUNCH. - If card punching is supposed to occur in the next cycle and no card will be present beneath the punch die, a B Fault computer stop is produced and the NO CARD IN PUNCH indicator on the Card Unit Control cabinet glows. This fault may be caused by a feed failure or by errors in the computer program. When the source of error is removed, the computer program must be restarted from the beginning.

(f) PUNCH JAM. - If a card jams under the edge of the punching die, the card equipment stops, all voltages to it are dropped, and the small red JAM indicator on the Card Unit cabinet glows. If this occurs, shut off the equipment power.

(g) STOP. - A stop light on the Card Unit Control cabinet glows and the card equipment stops if any of the following occur.

1 Read stacker full.

2 Write stacker full.

3 Read feed hopper empty.

4 Punch feed hopper empty.

5 The Stop button is pressed.

6 STANDBY switch is thrown to forward position (away from operator)

7 A Punch Jam occurs.

Note that all of these stops except 7 permit resumption of operation without loss of data. For stops caused by 3 or 4, move the STANDBY switch to the forward position during refill of input hoppers. (During the manual operation which clears the channels, the depression of the CLEAR button causes a bypass of the circuitry which normally causes an equipment stop when the feed hoppers are empty.) Note that during card reading or card punching operations, the hopper not being used for such operations must contain at least one card to prevent the occurrence of the stop caused by 3 or 4 above.

6. UNIVAC LINE PRINTER.

a. GENERAL. - The Univac Line Printer equipment, in conjunction with the IOB register and associated IOB control circuitry, provides a means of output from the Univac Scientific Computer. The Univac Line Printer equipment, enclosed in a single cabinet (shown in Figure 7), comprises a Format Switchboard, an External Control Panel, a carriage and platen assembly, print wheels, and associated control mechanisms. The medium of information representation is printed paper on which the printed characters may be decimal numbers, letters, a period, and a minus sign. The paper is positioned in the printer on a carriage similar to a typewriter carriage. The characters chosen to be printed on one line across the width of the paper are printed almost simultaneously, with the design of the Line Printer allowing the printing of 150 such lines per minute. The spacing between lines is automatic, if so desired, during continuous printing of lines and may be manually set for one, two, or three spaces. In addition, paper shifts (without printing) of multiples of three, six, and nine spaces may be initiated by a single computer instruction. Each line may have any of the allowable characters printed in any of 92 columnar positions. Each column has an associated print wheel on which 35 characters are available for printing. Each of the 34 numbers and letters below are present on each print wheel. The period is present on the even-numbered print wheels 2, 4, 6, etc., and the minus sign is present on the odd-numbered print wheels 1, 3, 5, etc., thus allowing a period and a minus sign to be printed in every other columnar position on a line. The characters to be printed in given columnar positions on a given line are chosen according to an associated 11-row, 92-column coded image as assumed in the computer.

An image has the following coding for available characters.

ROW	CHARACTER																																		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R	S	T	U	V	W	X	Y	Z	— or •
11											1								1								1								1
0	1											1								1								1							
1		1											1								1								1						
2			1											1								1								1					
3				1											1								1								1				
4					1											1								1								1			
5						1											1								1								1		
6							1											1								1								1	
7								1			1	1	1	1	1	1	1	1																	
8									1										1	1	1	1	1	1	1	1									
9										1																	1	1	1	1	1	1	1	1	

Note: Blank spaces denote zeros in the image as stored in the computer.

The absence of "1's" in a column of the image results in a space in the corresponding position on the paper; or, if a mechanical setting is made on the printer, a zero is printed in this position, provided that a print selection was made for the position immediately to the left. A maximum of approximately nine consecutive zeros in a group may be printed in this fashion.

b. PROGRAMMING FOR OUTPUT. - Information is transmitted to the printer in response to programmed External Function and External Write instructions. Each row of the image is formed in the computer by positioning at three addressed computer locations the bits for that row which are part of the code of the characters to be printed. Thus, 33 addressed locations are required to represent an image. The image is divided into three fields: Field I consists of the first or left-most 36 columns; Field II, the second 36 columns; and Field III, the third or right-most 20 columns. Three External Write instructions, which must be programmed consecutively, transmit the coded information of each row of the image to the IOB Register, or, as follows.

EWjv $(v) \longrightarrow X$, $(X) \longrightarrow IOB$; $(v_{35} \ldots v_0)$ being Field I of Row r

EWjv $(v) \longrightarrow X$, $(X) \longrightarrow IOB$; $(v_{35} \ldots v_0)$ being Field II of Row r

EWjv $(v) \longrightarrow X$, $(X) \longrightarrow IOB$; $(v_{35} \ldots v_{16})$ being Field III of Row r

The three instructions are repeated for each of the 11 rows, the rows being transmitted in the order r = 9, 8, ... 1, 0, 11. All three instructions must be executed even though there may not be information in all three fields to be printed.

Between successive External Write instructions, the current content of IOB must be transmitted to the Line Printer, allowing IOB to be cleared. Programming a properly coded External Function instruction to precede the External Write instructions initiates the operation of the Line Printer, allowing it to receive information from IOB. Thus, for controlled operations of the Line Printer, the External Function instruction places "ones" in selected stages of IOB. The stages selected, with the operations of the Line Printer which are initiated by the EF instruction and the presence of ones in these stages, are listed below.

IOB_{34} - Start (master bit)
Causes one cycle of operation of the Line Printer and the automatic advance of the paper 1, 2, or 3 spaces unless the Stop Paper or the Jump selection was also made.

IOB_{13} - Jump
Causes the paper to advance nine spaces during a cycle of operation. (No printing permissible during this cycle.)

IOB_{12} - Skip
Causes the paper to advance according to the Format Switchboard settings. (A Print selection is not permissible in an EF instruction which includes the Skip bit.)

IOB_{11} - Interrupt

IOB_{10} - Print
Permits one line of characters (an 11 row image) to be sensed and printed during a cycle of operation.

IOB_9 - Stop Paper
Stops the automatic advance of the paper during a cycle of operation.

Each cycle of operation of the Line Printer is divided into 15 points (9, 8, ..., 1, 0, 11 ... 14, 15), each point being 26.67 milliseconds in duration. During each of the 11 points (9, 8, ..., 1, 0, 11) of the cycle, a row of the image is sensed by the printer provided three External Write instructions are executed during this time and the IOB Print (and IOB Start) selections were made by a previous External Function instruction. If these selections are in effect, the series of transmissions to the Line Printer effected by the 33 External Write instructions positions the chosen print wheels. The characters in position for printing are printed on paper sometime during or after the cycle point six. The automatic paper advance occurs before cycle point six. The number of spaces the paper is automatically advanced is determined by a switch setting on the Format Switchboard which is discussed later.

Time factors that must be taken into consideration in programming for Line Printer output are pointed out in the following program outline. A time lag due to inertia when the printer is initially started may occur which will delay the execution of the first point of the cycle of operation. Note that an External Write of Field I for Row 9 should immediately follow the External Function instruction which starts the printer and, for Rows 8,... 0, 11, should occur before or during the early part of the corresponding ten points. Other computer instructions may be programmed and executed between cycles and during cycle points in the remaining available computer time.

To advance the paper and print one line

```
EF-v                 (v) = Start Line Printer
                           Print

     Within 1.5 ms of the beginning of the point the first EW ⎫
     must be executed                                         ⎪
                                                              ⎪
EWjv                   Write Field I                          ⎬
EWjv                   Write Field II                         ⎪     point 9,
EWjv                   Write Field III                        ⎪     26.67 ms
Other instructions                                            ⎭
```

```
    Within 1.5 ms of the beginning of each point, the first
    EW must be executed

EWjv                Write Field I
EWjv                Write Field II                          Repeat for
EWjv                Write Field III                         points 8, 7
Other instructions                                          ... 0;  each
 '                        '                                 26.67 ms
 '                        '
 '                        '

    Within 1.5 ms of the beginning of the point, the first
    EW must be executed

EWjv                Write Field I
EWjv                Write Field II                          point 11,
EWjv                Write Field III                         26.67 ms
Other instructions

Other instructions                                          points 12, 13
                                                            14, 15, each
                                                            26.67 ms
```

Continuous printing of lines may be programmed by repeating the preceding program the desired number of times with the instruction initiating each repeating cycle being programmed during point 12 of the preceding cycle. Thus, for continuous operation of the Line Printer, the External Function instruction to initiate a new cycle must be executed within 50 milliseconds of the Write Field III instruction for row 11.

The selection of the Interrupt bit, IOB_{11}, facilitates programming for output to the Line Printer. A selection of IOB_{11}, in the External Function instruction in which IOB_{34}, Start, and IOB_{10}, Print, are also selected energizes an interrupt line in the printer so that an interrupt signal is sent to the computer (interrupt) control each time a row of the card image is to be sensed, i.e., at the beginning of cycle points 9, 8, ...,1, 0, 11. The first External Write instruction must be executed within 1.5 milliseconds after the interrupt signal is sent to the computer.

The interrupt selection may also be made by setting the interrupt switch on the Line Printer to its ON position. With this setting the interrupt line in the printer is energized as above when a Start and Print selection is made.

c. FORMAT SWITCHBOARD. - By making the appropriate switch settings on the Format Switchboard, the printing of lines in accordance with a chosen format may be accomplished without executing External Function instructions to initiate individual spacing operations of one, two, three, or nine lines. Skips of one, two, three, six or nine spaces during one printer cycle, or consecutive skips of these spacings during successive printer cycles, may be initiated by a single computer instruction, after which the computer is free to execute any other instructions.

The ten columns of switches on the Format Switchboard provide a format "count" control of available spacing operations. A spacing operation depends upon the switch settings in the column corresponding to the format count.

The functions of the three portions of the switchboard: Control, Space, and Multiply, and their switches, are explained below.

(1) CONTROL. - A switch in the Hold setting holds computer control of spacing in effect until an External Function instruction containing Start and Skip IOB Select bits is executed and the next format count position has an Advance switch setting. A printer cycle of operation initiated by such an EF instruction is necessary to make the transition from computer control to format control. A switch in the Advance setting, in conjunction with Skip and Start bits in the preceding EF, releases computer control of spacing, advances the paper in the printer according to switch settings on the Space and Multiply portions of the switchboard, and advances the format control count to the next position. Adjacent switch settings of Advance in the Control portion will automatically continue the format control count across the switchboard (to the next count position with a Hold switch setting) with spacing occurring according to the corresponding switch settings in Space and Multiply. (Spacing occurring for a count position which has an Advance switch setting is defined as being under "format spacing control".) A switch in the Clear setting advances the format control count to the next count position disregarding any switch settings in the Space and Multiply positions.

Switch settings of Advance or Clear in the first format control count position, i.e., "home" count position, are not effective as described above. This position has effectively an "automatic" Hold switch setting.

(2) SPACE. - Spacing between lines is affected by settings on the Format Switchboard during both computer controlled and format controlled spacings. Under computer control the paper is advanced the number of spaces according to the switch settings in the Space portion only of the Switchboard. Under format spacing control the paper is advanced (with the exception noted below) the number of spaces according to the switch settings in both the Space and Multiply portions of the switchboard.

(3) MULTIPLY. - Under format spacing control the paper is advanced one, two, or three spaces (according to the Space setting) if a Multiply switch is in the OFF position (unless a Stop Paper bit is in effect); or three, six, or nine spaces (each space setting multiplied by three) if a Multiply switch is in the ON position.

Note that the change from computer control to format spacing control requires one printer cycle, i.e., advancing from a Hold switchboard setting to the first Advance switchboard setting requires a cycle of operation instigated by an EF instruction with the Start and Skip bits. During this cycle of operation, the paper is advanced automatically one, two, or three spaces. This spacing action must be considered in making switchboard settings in the Advance switchboard position in order to effect the desired overall skip of spaces. The Stop Paper bit could be programmed in the EF instruction which causes the change to format control. However, if this is done, the Stop Paper bit effects paper shift action under format spacing control (until the next Hold setting) as follows: if the Multiply switch is OFF, no paper spacing results for that particular format count position; if the Multiply switch is ON, a paper skip of three, six, or nine spaces results for that particular count position.

The following listings show optional spacings available under computer control and format control of spacing. If no spacing is to occur under format spacing control, the format control count should be in the home count position. (This is insured by depressing the Clear Control button for at least two seconds.) The format switchboard settings under computer control below are for this home position or any count position which has a Hold switch setting.

Under Computer Control

			FORMAT SWITCHBOARD SETTINGS		
	FUNCTION	IOB SELECT BITS	CONTROL	SPACE	MULTIPLY
1.	Start, advance paper, and print (if desired)	Start Print (if desired)	Hold	1,2,or 3	OFF*
2.	Start, stop advance of paper, and print (if desired)	Start, Stop Paper Print (if desired)	Hold	1,2,or 3*	OFF*
3.	Start and jump 9 spaces without printing	Start, Jump	Hold	1,2,or 3*	OFF*

Note that before printing the first line, the paper must be adjusted one, two, or three spaces ahead of the position in which the first printing is desired unless the EF instruction contains a Stop Paper bit.

Under Format Spacing Control

			FORMAT SWITCHBOARD SETTINGS		
	FUNCTION	IOB SELECT BITS**	CONTROL	SPACE	MULTIPLY
4.	Single, Double, or Triple Space, advance format control count to next position.	Start, Skip	Advance	1,2,or 3	OFF
5.	Space six or nine, advance format control count to next position.	Start, Skip	Advance	2 or 3	ON
6.	No spacing, advance format control count to next position.	Start, Skip	Clear	1,2,3*	OFF*

* These switch settings are not effective in setting up the function.

** Included in a previous EF used for the transition from computer controlled operation to format spacing control. The effect of a Stop Paper bit included in this instruction is not shown here.

Functions 1, 2, 3, 4, and 5 require timewise one print cycle.

Function 6 settings continuously across the board require a maximum of one second to clear to the "home" position.

An optional format for Line Printer output is illustrated in Figure 8. The form length used is 66 spaces, generally the number of spaces on the folded pack paper commonly used. The following listing shows the selections necessary to achieve the printing of a page in this format. (Each External Function instruction must be followed by the appropriately coded External Write instructions.) In this case a Stop Paper bit was included in the EF instructions which effect the change from computer control to format control. Since the paper skips desired under format control are multiples of three, the Multiply switch in the Advance count positions is ON; hence the stop paper bit in effect will not hinder the desired paper spacing action from occurring.

FORMAT CONTROL "COUNT"	FORMAT SWITCHBOARD SETTINGS CONTROL	SPACE	MULTIPLY	SPACES ADVANCED	IOB SELECT BITS (In a total of 27 coded EF-v instructions)
1	HOLD	1	OFF	18	Start, Print (repeat this coded EF 18 times)
					Start, Stop Paper, Skip
2	ADVANCE	3	ON	9	
3	ADVANCE	3	ON	9	
4	HOLD	3	OFF	21	Start, Print (repeat this coded EF 7 times)
					Start, Stop Paper, Skip
5	ADVANCE	3	ON	9	
6	CLEAR	1	OFF	0	
7	CLEAR	1	OFF	0	
8	CLEAR	1	OFF	0	
9	CLEAR	1	OFF	0	
10	CLEAR	1	OFF	0	

Total of 66 spaces

d. OPERATION. - The procedures listed below should be followed to initiate use of the Univac Line Printer.

(1) Set the Main Power switch on the External Control Panel to ON.

(2) Check that the Stop switch on the External Control Panel is OFF.

(3) Depress the Clear button on the External Control Panel.

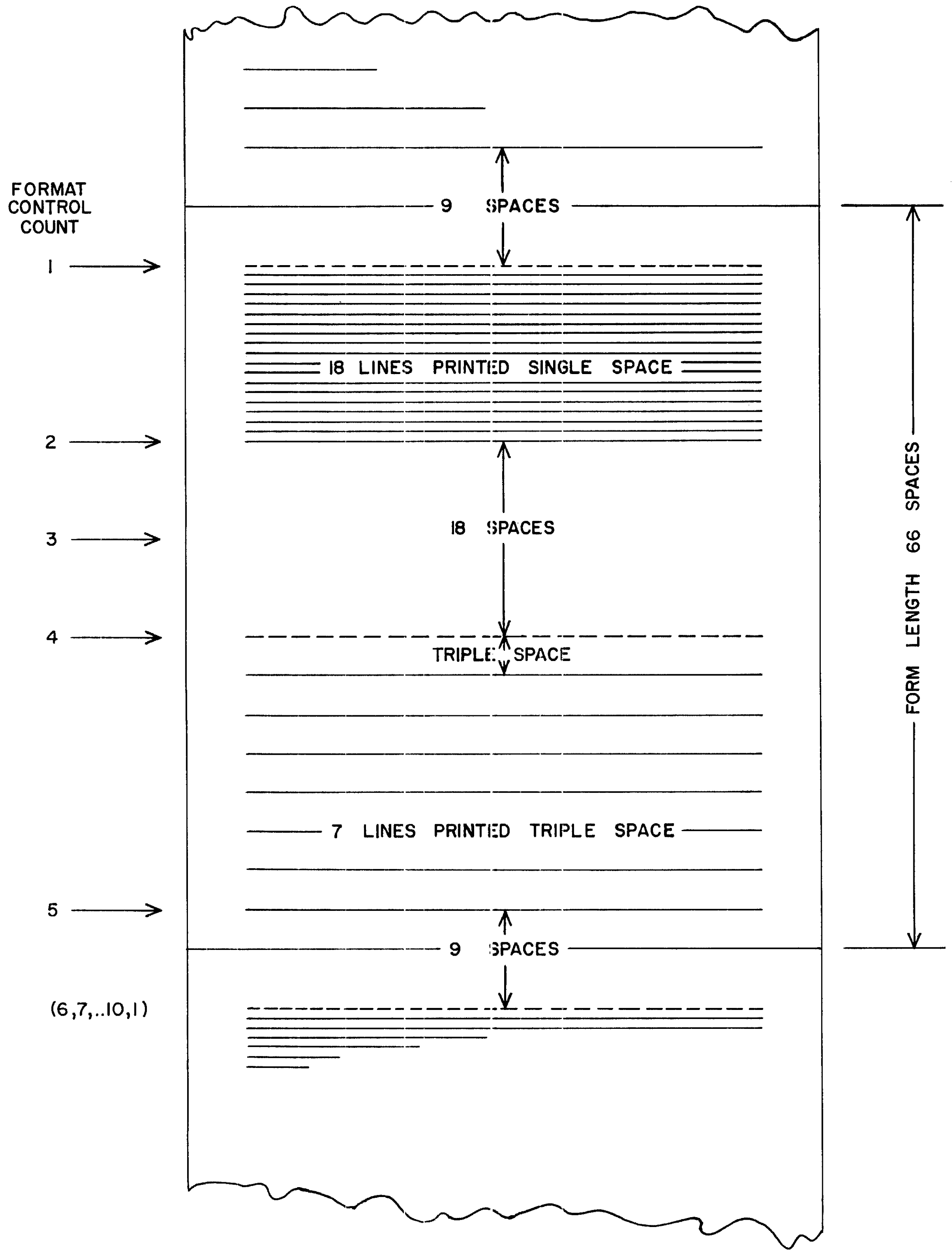


Figure 8. Optional Format for Line Printer Output

(4) Depress the Clear Control button on the Format Switchboard for at least two seconds.

(5) Insert paper in the platen assembly.

The following conditions require that the Clear Control on the Format Switchboard be depressed for at least two seconds to insure that the format control count is back in the home position. These conditions are as follows:

(1) Paper is being changed

(2) A fault occurs

(3) The format is changed

(4) An improper program is scheduled.

During the operation of the Line Printer four types of faults may occur which are indicated visually by a light on the printer or a FAULT indication on the Supervisory Control Panel of the computer. These faults are as follows:

(1) NO PAPER FAULT. - If the paper in the Line Printer is almost depleted, a fault circuit stops the computer and gives a No Paper fault indication on the printer control panel. Pressing the Clear button on the printer clears the fault. Starting computations anew will initiate a second fault condition. Since a No Paper fault occurs slightly before the end of the paper has passed the printing position, several more lines, depending on the spacing, may be printed. This fault becomes effective after the current cycle is completed, thus no information will be lost.

(2) MECHANICAL JAM FAULT. - If the printer stops before completing a cycle, a B Fault stops the computer and printer and is indicated by the IO light on the Supervisory Control Panel. There is a possibility that this fault may be cleared by pressing the Master Clear button on the computer.

(3) NO INFORMATION FAULT. - If IOB does not contain the Field I portion of a row (i.e., the first EW instruction has not been executed) within 1.5 milliseconds after the start of a cycle point, a B Fault occurs. The Line Printer completes the cycle before stopping but will print erroneously since no further EW instructions occur within the cycle.

(4) OVERHEAT FAULT. - This fault will cause an A Fault computer stop. Operations may be resumed temporarily without loss of information by setting the BYPASS INTERLOCK switch on the Supervisory Control Panel, pressing the CLEAR A FAULT button, and pressing the START button on the computer.

7. MAGNETIC TAPE SYSTEM

a. GENERAL DESCRIPTION. - The Magnetic Tape System of the Univac Scientific Computer System comprises a number of Uniservo tape handling mechanisms, which are located externally to the computer, and an electronic control section which is located in the computer structure. The number of Uniservos used is optional up to a maximum of ten functional units. By means of manual selections the unit designations may be assigned in any manner to the functional units. Use of the Uniservo units makes possible off-line processing of information by a variety of Univac peripheral equipments.

For input to the Univac Scientific, information may be recorded on tape in three forms:

Fixed Block Length Recording
Variable Block Length Recording
Continuous Data Input Recording

Output information from the Univac Scientific may be recorded on tape in the Fixed Block Length Recording form and the Variable Block Length Recording form.

The Fixed Block Length mode is standard with the Magnetic Tape System. Optional Control circuitry may be added to provide both the Variable Block Length mode and the Continuous Data Input mode.

The Variable Block Length mode reads and records information on tape in blocks of variable length. A block of information is recognized by a one-inch space preceding it and following it in which no information is recorded. The length of the block is limited only in that the data input from it must not exceed the capacity of high speed storage.

The Continuous Data Input mode reads information recorded continuously on the tape with the only limitation on the length of a "block" of information being the length of the tape. This form of recording is useful for real time observations which will not permit interruptions to format the information in fixed or variable block lengths. Data input from tape recorded in this manner would need be interrupted when the capacity of high speed storage is reached.

The Fixed Block Length mode reads and records information on the tape in blocks of fixed length. The remainder of the remarks in this description apply in particular to this mode of operating.

(1) TAPE CHARACTERISTICS. - Unitape is the metallic tape used by the tape handling equipment in recording and reading information. Information is recorded as magnetized areas in eight channels across the width of the tape. Data bits are recorded in six of these channels; one channel contains parity check bits on the six data channels; and one channel contains sprocket, or timing, signals.

A Uniservo is a unit of tape handling equipment. A Uniservo comprises a read/write head, an erase head, a bad spot (in the tape) detector, and tape handling mechanism such as the tape reel mount, reel drives, etc. In a reading operation the read/write head detects "1's" recorded on the forward or backward moving tape; in a writing operation the read/write head records both "0's" and "1's" on the forward moving tape. During a writing operation the erase head is also activated such that the tape in its passage through the erase head has "0's" written on it. The bad spot detector enables the Uniservo to interrupt reading, moving, or writing operations until the undesirable tape area passes the read/write head.

A tape speed of 100 inches per second is standard. The approximate length of a reel of tape is 1500 feet.

A column of eight binary digits across the width of a tape is termed a line; the six data binary digits, a hexabit character. Lines are recorded on the tape at a density of 128 lines per inch (standard) or 50 lines per inch.

A block consists of 720 consecutive lines; a blockette consists of 120 consecutive lines. A "dead space" (in which no information is recorded) of 1.0 inch (standard) or 2.4 inches exists between blocks; a dead space of zero, 0.1, or 1.0 inches exists between the six blockettes of a block. The optional recording formats are selected by program control in accordance with the intended future use of the recorded tape, possibly with Univac auxiliary equipment.

When a stop of tape movement occurs, the tape is halted in the dead space between blocks. Blockettes and blocks are recognized by the tape control section by counting the timing signals recorded on the tape. A timing device, actuated in the last blockette, signifies the end of a block if the interval between timing signals exceeds a certain length of time. The detection of the end of a block in combination with a stop signal causes a halt of tape movement.

In terms of a 36-bit computer word, six lines are necessary for recording one word. A block comprises 120 computer words; a blockette comprises 20 computer words. Thus, approximately 326,000 computer words may be stored on a 1500-foot reel of tape. The maximum transfer rate of information between the computer and the tapes is approximately 1810 words per second. This is assuming a free-running tape, one-inch block spacing, and zero blockette spacing. Average magnetic tape times for other block and blockette spacings are shown in Table 7.

(2) PROGRAMMING TAPE OPERATIONS. - A programmed External Function instruction (17-V) initiates the various reading, writing, and positioning operations of the Uniservos. Coded information provided by the External Function instruction includes the following:

(a) specification of optional recording mode if available
(b) designation of the selected Uniservo
(c) type of tape operation to be performed
(d) type of recording format for writing, i.e., block and blockette spacing and recording pulse density.
(e) number of blocks to be moved <u>without</u> reading or writing.

The External Function instruction transfers the content of its v-address to the Input/Output Register, IOB. The 36-bit word thus introduced into IOB designates the magnetic tape operation to be performed. The "Select Magnetic Tape"

TABLE 7

AVERAGE MAGNETIC TAPE TIMES

To estimate running times of programs using magnetic tapes

Recording Density (lines per inch)	Block Space (inches)	Blockette Space (inches)	Block Length (including block space, in inches)	Block Period* (milliseconds)	Rate (36-bit words per second)
128	1	none	6.625	66.25	1811
128	1	0.1	7.125	71.25	1684
128	1	1.0	11.625	116.25	1032
128	2.4	none	8.025	80.25	1495
128	2.4	0.1	8.525	85.25	1408
128	2.4	1.0	13.025	130.25	921
50	1	none	15.4	154.0	779
50	1	0.1	15.9	159.0	755
50	1	1.0	20.4	204.0	588
50	2.4	none	16.8	168.0	714
50	2.4	0.1	17.3	173.0	694
50	2.4	1.0	21.8	218.0	550

Tape speed: 100 inches per second
Time to reverse direction of tape: 600 milliseconds

* These block periods are for either free run or for a single block. The block period does not include the starting or stopping times.

bit in IOB causes a transmission of the content of IOB to registers in the tape control system. The tape control system then interprets this word and initiates tape operation accordingly. A list of IOB Select bits which govern the tape operations is given in Table 8.

If a reading or writing operation is initiated, the External Function instruction which initiates the operation must be followed by an appropriate number of External Read (76jv) or External Write (77jv) instructions to transfer the information between the IOB register and the computer memory. One hundred twenty of these External Read instructions or External Write instructions are needed for each block of tape. The reading or writing operation is terminated by an External Function instruction specifying a "Stop" code. This is executed after the last word in the final block has been read or written. When a single block is to be read or written, the "Stop" code may be included in the External Function instruction which initiates the read or write.

A list follows of tape operations initiated by an External Function instruction.

READ FORWARD. - Read data from tape on the specified Uniservo, assemble into 36-bit computer words, and transfer the words to the computer Input/Output Register, IOB.

The initiation of this operation causes the designated Uniservo to read a number of blocks from the tape. The reading operation must be terminated by an External Function Stop instruction which is programmed immediately following the External Read instruction used to read the last word in the final block. If it is desired to read one block only, the "Stop" code may be included in the External Function Read Forward instruction which precedes the 120 External Read instructions.

WRITE FORWARD. - Transfer words from the computer Input/Output register, IOB, and record on tape on the specified Uniservo. Each word is recorded in six segments of six bits each in accordance with the specified density and spacing.

The initiation of this operation causes the Uniservo to write a number of blocks on the tape. The writing must be terminated by an External Function Stop instruction used to write the last word in the final block. If it is desired to write one block only, the "Stop" code may be included in the External Function Write Forward instruction which precedes the 120 External Write instructions.

READ BACKWARD. - Identical to Read Forward except that the tape is moved in the reverse direction. The bits of each computer word are assembled in the same order as in a Read Forward operation.

MOVE FORWARD (n BLOCKS). - Move tape forward n blocks on the specified Uniservo, without a read or write operation ($0 \leq n \leq 2^{12}-1$).

MOVE BACKWARD (n BLOCKS). - Move tape backward n blocks on the specified Uniservo, without a read or write operation ($0 \leq n \leq 2^{12}-1$).

TABLE 8. EXTERNAL FUNCTION BIT ASSIGNMENT FOR UNISERVO CONTROL

The bits in IOB_{23} ... IOB_{12} are placed in the Tape Control Register (TCR_{11} ... TCR_0); the bits in IOB_{11} ... IOB_0 are placed in the Block Counter (BK_{11} ... BK_0) where they are used to govern the number of blocks moved in a Move Forward or Move Backward operation.

IOB_{31}	= 1 - Select Magnetic Tape
IOB_{23}-IOB_{22}	Select Rewind or Stop Tape = 01 - Rewind = 10 - Rewind Interlock = 11 - Stop
IOB_{21}	Select Block Spacing = 0 - 1 inch interblock spacing = 1 - 2.4 inch interblock spacing
IOB_{20}-IOB_{19}	Select Blockette Spacing = 00 - zero spacing = 01 - 0.1 inch spacing = 10 - 1.0 inch spacing
IOB_{20}-IOB_{19}	= 11 - Select operation in Variable Block Length mode or Continuous Data Input mode*
IOB_{18}-IOB_{16}	= 100 - Select option of operating in Continuous Data Input mode or change to option of operating in Variable Block Length mode.**

* If the Variable Block Length and Continuous Data Input modes are available, these bits included in an EF for an allowable tape operation cause the mode of operation to be either (1) Variable Block Length instead of Fixed Block Length, or (2) Continuous Data Input instead of Fixed Block Length. A computer Master Clear automatically establishes the conditions necessary for the choice described in (1).

** An EF with these bits and the Select Magnetic Tape bit establishes the conditions necessary for the choice described in (2) above if the conditions for (1) were established previously; and vice versa.

TABLE 8. EXTERNAL FUNCTION BIT ASSIGNMENT FOR UNISERVO CONTROL

IOB_{18}-IOB_{16}	Select Tape Operation = 001-Read Forward = 010-Move Forward = 011-Write Forward (128 lines/inch) = 101-Read Backward = 110-Move Backward = 111-Write Forward (50 lines/inch)
IOB_{15}-IOB_{12}	Select Uniservo unit or Read Bias Level = 0001-Uniservo 1 = 0010-Uniservo 2 . . . = 1010-Uniservo 10 = 1101-Normal read bias = 1110-Low read bias = 1111-High read bias
IOB_{11}-IOB_0	Select the number of blocks to be moved in a Move Forward or Move Backward operation.

STOP. - Stop tape movement and tape reading or writing after the desired number of blocks have been read or written.

An External Function Stop instruction must be programmed immediately following the terminal External Read or External Write instruction used to read or write the last word in the final block of a group of blocks. This External Function instruction need not, and in fact should not, specify a particular Uniservo.

REWIND. - Rewind tape on the specified Uniservo to the leader position.

REWIND INTERLOCK. - Rewind tape on the specified Uniservo to the leader position; and provide an interlock which prevents further effective references to that tape unit until appropriate steps are taken to remove the interlock.

No more than one Uniservo can be in operation at any one time unless the operations are Rewind and Rewind Interlock. After either of the rewind operations is initiated, the Uniservo proceeds under its own control until the operation is completed; therefore, an operation on another unit may be performed before rewinding is completed. Any number of functional units may be rewinding concurrently.

CHANGE BIAS. - Change the read bias level to higher or lower than normal; or return bias level to normal if high or low bias level was chosen by a previous External Function. No uniservo specification is necessary or possible in the same instruction.

b. OPERATION THEORY.

(1) MAGNETIC TAPE CONTROL. - The transfer of information between the computer and the magnetic tapes is controlled by the Magnetic Tape Control system situated within the computer. The presence of an IOB_{31} Select "1" bit, as placed in IOB by an External Function instruction, informs the tape control system that a tape operation is desired and the contents of IOB are to be transferred to the Tape Control Register (TCR). The presence of the proper tape operation codes in TCR provides the control system with the information needed to carry out the particular tape operation designated. These tape operations are Write Forward, Read Forward, Read Backward, Stop Tape, Move Forward, Move Backward, Rewind, Rewind with Interlock, and Change Bias.

A properly recorded block consists of 720 lines (each line containing six data bits) or 120 computer words. Thus, 120 External Writes or External Reads need to be executed to record a block or read a block. The link between IOB and the tape is a 36-bit Tape Register (TR) in the tape control section. A writing operation involves the transmission of a word from IOB to TR, the breakdown in TR of the 36-bit word into six hexabit characters, the generation of a parity check bit for each character, and the transfer of each character, its parity check bit, and a timing signal (or sprocket pulse) to a line of tape. After six lines of tape are written, the next word to be written is transmitted to TR. Since the tape operations proceed under their own timing control, a word must be ready in IOB for transfer to TR each time tape control signifies it is ready to write a word. If an External Write instruction has not been executed to fill IOB, a No Information fault is generated which causes a B Fault computer stop and a tape stop at the end of the current block. Thus, a No Information fault indicates that insufficient EW's are programmed, or an External Write is programmed to be executed too late.

The parity check bit generated during a writing operation is a "0" if the number of "1's" in the character is odd, and the parity check bit is a "1" if the number of "1's" in the character is even. This will always result in an odd sum if the character and parity bits are added together.

The timing signals are recorded on the tape at the same time as the lines of information. During a writing operation, previously recorded signals are removed from the tape by the erase head which "erases" the entire width of the tape during the tape passage under it. The position of the erase head is several inches in advance of the read/write head. This insures the removal of signals from inter-block spaces during continuous writing from the beginning of the tape. However, it is not possible to guarantee a safe rewrite of a previously recorded block when the writing operation is initiated with the read/write head positioned in the inter-block space immediately preceding that block. Also, a block in the middle of a tape cannot be rewritten without erasing at least part of the following block. The last block that has been recorded may be rewritten, but the entire block will not be subjected to the erase head when the read/write head is positioned before that block. At any point writing can be stopped, the tape moved backward for reading purposes, and writing later resumed safely at the point at which it was stopped. In particular, a block can be written and then read backward to check for accuracy in writing. If no errors are detected, writing can be continued by repositioning the tape after the block.

Thus, any writing operation should be started either at the beginning of the tape or at the point at which previous writing was stopped.

A reading operation involves the assembly of data from the tape in the Tape Register, a parity check on each line of tape as it is received in TR, and the transmission of a 36-bit word from TR to IOB. The flow of information in a reading operation is from tape to the Align Input Register (AIR) to the Tape Register to IOB. The transmission from AIR to TR is caused by the sprocket pulse (delayed) which was recorded with the bits now in AIR. Information is assembled in TR six data bits at a time. When six lines have been read, R contains a complete word and tape control performs the transmission TR to IOB. Again, the assembly of words in TR proceeds timewise under tape operation control. The nature of the IOB lockout system provides the wait for IOB to be filled if an External Read instruction is begun before IOB receives information from external equipment; after this occurs the External Read is completed and the content of IOB is sent to computer storage. If tape control performs a second transmission TR to IOB before IOB has been cleared by an ER, an IO computer fault is incurred, a B Fault computer stop occurs, and tape movement is stopped at the end of the current block. This again could mean a lack of sufficient ER's programmed or the tardy execution of an External Read.

When a block of information is read from magnetic tape, a check is also made to determine if the proper number of lines are recorded in the block. A properly recorded block consists of 720 lines. A count is kept of the number of lines in a block by a series of counters which are advanced by the reception of the sprocket pulses (timing signals) recorded on the tape. If a 720 count has been tallied and another sprocket pulse is detected within a certain period of time, an inter-block space is not passing beneath the read/write head and a 720 check failure is indicated. Also, if less than a 720 but more than a 600 count has been tallied and no sprocket pulse is detected within a certain period of time, an inter-block space may be passing under the read head and a 720 check failure is indicated. A 720 check failure incurred in a reading operation stops the tape unit and causes a Sprocket Error fault indication which causes a B Fault computer stop.

When the parity check is made during the reading operation, a parity error is indicated by a "0" sum. The occurrence of a parity check error is detected at the end of the block in which it occurred. At this time a tape stop is initiated and a parity error is indicated by setting a "1" in stage IOA_0 of the IOA register. Thus, the content of IOA must be examined by the program immediately following the execution of the External Read instruction used to read the last word in each block. If no error indication is detected, the program proceeds normally. If an error is detected, a standard subroutine stored in the computer can be used under control of the main program to initiate and perform re-reading operations in an attempt to read the block correctly. The automatic stop of a tape unit on a parity error indication is effectively completed before the initiation of a re-read operation is allowed by the tape control system.

The features which provide a read backward operation and a change in the bias level may permit proper reading of marginal signals which have caused a parity check failure. Since both of these operations are available under program control, a correct reading of the block in question may be accomplished without a computer stop. A change in the read bias level to either higher or lower than normal may permit the correct reading of the block by virtue of ignoring any "noise" factor on the tape or picking up any marginally recorded bits.

The halt of a correctly executed read or write operation occurs when a stop code in the Tape Control Register (TCR) and an "end of block" count both exist. Hence, if only one block is to be read or written, the stop code may be placed in TCR by the External Function instruction which initiated the read or write operation; if more than one block is to be read or written, the stop code is placed in TCR by an EF instruction following the last EW or ER instruction. In this case, the EF instruction needs only to contain the stop code in its v-address. A programmed stop tape operation is necessary only to conclude reading and writing operations.

A programmed External Function instruction for a move operation must include the Move Forward or Move Backward code and a specification of the number of blocks to be moved. The number of blocks n to be moved are coded in IOB_{11} ...IOB_0. During a moving operation no information transfer occurs past the Align Input Register. The delayed sprocket pulse from each line of tape does not cause the transmission AIR to TR but is used only to form a count of the lines moved. Each time a block count is reached, the Block Counter is reduced by one. When BK has been reduced to zero, a tape stop is initiated.

A 720 check failure incurred in a moving operation stops the tape unit and causes a sprocket error indication and a B Fault computer stop if a $>$ 720 count is detected; if a $>$ 720 count is detected, the 720 check failure is ignored in that no tape stopping action or error indication is effected.

A change in the bias level needs to be programmed only when an incorrect reading operation has occurred. Reading forward and backward at the high and low bias levels may accomplish a correct reading of the block. It is not permissible to program any other tape operation in the instruction which specifies a change in bias. The return to the normal reading bias level must also be programmed unless a computer Master Clear occurs which also accomplishes this.

Both the Rewind and Rewind with Interlock operations cause the tape on the Uniservo specified to be positioned to its leader position. A Uniservo whose tape has been rewound with interlock cannot be referenced effectively until the Uniservo door interlock switch has been opened and closed. This occurs when the Uniservo is provided with another tape.

After the Rewind and Rewind with Interlock operations are through their initiation phase, another tape operation on a different Uniservo may be started. Thus, any number of Uniservos may be rewinding concurrently. If a tape operation is desired on a Uniservo which is rewinding when this tape operation is initiated, the tape operation is held up until the rewinding is completed. Then, if the tape has not been rewound with interlock the tape operation is resumed and completed.

It is not possible to execute correctly a second tape operation on any Uniservo while a previous one is still in progress unless the tape operation in progress is a rewind operation.

(2) GLOSSARY. - The following glossary lists terms pertaining to the magnetic tape system. A brief description is given of the primary function of the principal registers and counters involved in tape control.

Figures 9, 10, and 11, following the glossary, illustrate the use of the registers and counters and the sequence of events, in write, read, and move operations.

Read/Write head - Binary digits are represented in channels across the width of the tape as areas magnetized in opposite directions. In a writing operation areas in the channels on the forward moving tape are magnetized in the "0" direction except when pulses are received by the read/write head from tape control in which case areas are magnetized in the "1" direction.

In a reading operation the read/write head, as it detects in the channels the areas magnetized in the "1" direction, directs pulses to be sent to tape control. Accurate detection occurs when the backward or forward moving tape has reached its free-running rate (100 inches per second).

Erase head - during a writing operation, the erase head magnetizes in the "0" direction the entire width of the tape in its passage under the erase head. The position of the erase head several inches from the read/write head is such that the traversal of the tape in a forward direction is first under the erase head and then under the read/write head. This guarantees a "clean" portion of the tape on which to write.

Tape Leader - the plastic length of the tape which precedes the metallic length of the tape on which information can be recorded.

Ringed tape - A tape reel fitted with a ring which prevents writing on this tape and in so doing provides indication of this condition. This feature is to be made available in the near future.

Writing Oscillator - emits pulses at the rate of one approximately every 80 μs (for writing 128 lines/inch) or one every 200 μs (for writing 50 lines/inch). The rate depends upon the selection made for density of lines in a writing operation. An oscillator pulse (in conjunction with other conditions) initiates the writing of a line.

Sprocket Pulse - formed by sensing the sprocket channel as a line of tape moves past the read/write head in a reading or moving operation. After a short delay the sprocket pulse is instrumental in a reading operation in routing the information read from that line of tape to the computer. In a moving operation each sprocket pulse is used to form a count of lines moved.

Parity bit - recorded on each line of tape during a writing operation and used as a check on the accuracy of reading each line of tape. The parity bit generated during writing a character (six data bits) is a "1" if the number of "1's" in the character is even and a "0" if the number of "1's" is odd. The sum of the data bits and the parity bit should be odd.

Bad Spot Control
Bad Spot on tape - Areas on the tape on which no recording should be attempted are marked as "bad spots". Holes punched in the tape preceding, following, and in the bad spot area are sensed by photoelectric tape readers during tape movement. During a reading operation, Bad Spot Control interprets the position of these holes and temporarily stops any transmissions from tape until the bad spot is passed; during a moving operation Bad Spot Control stops temporarily the counting procedure of the lines moved; during a writing operation, Bad Spot Control temporarily stops oscillator pulses from initiating the writing of a line.

Reading Bias - the voltage applied to the read/write head when reading from the tape. By changing the bias level to higher or lower than normal, the read/write head responds to weaker than normal signals on the tape or ignores objectionable "noise" factors on the tape.

Tape Control Register (TCR) - a 12-bit register in the tape control system which receives the bits in $IOB_{23} \ldots IOB_{12}$ when the bit in IOB_{31} is "1" (Select Magnetic Tape bit). The actual transmission IOB to TCR is held up (i.e., TCR is "locked out") until the tape control system signals that it is ready to receive another operation code. The presence of operation bits in TCR enables the various tape operations.

Block Counter (BK) - a 12-bit register in the tape control system which receives the bits in $IOB_{11} \ldots IOB_{0}$ on an IOB to TCR transmission as described above. The content of BK is used to regulate the number of blocks (n) moved in a Mo e Forward or Mo e Backward operation.

Tape Register (TR) - a 36 bit register in the tape control system through which words are routed in their transmission between tape and IOB. In a writing operation six-bit segments of a word in TR (placed there by an External Write instruction) are positioned in $TR_{5...0}$ and consequently written on tape. In a reading operation, a word is assembled in TR until six lines have been received; the transmission TR to IOB then occurs.

Align Input Register (AIR) - a seven bit register which receives six data bits and the parity bit from a line of tape passing under the read/write head in a read or move operation. In a reading operation the delayed sprocket pulse formed by sensing the sprocket channel in the same line of tape causes the transmission $AIR_{5...0}$ to $TR_{35...30}$. AIR is then cleared.

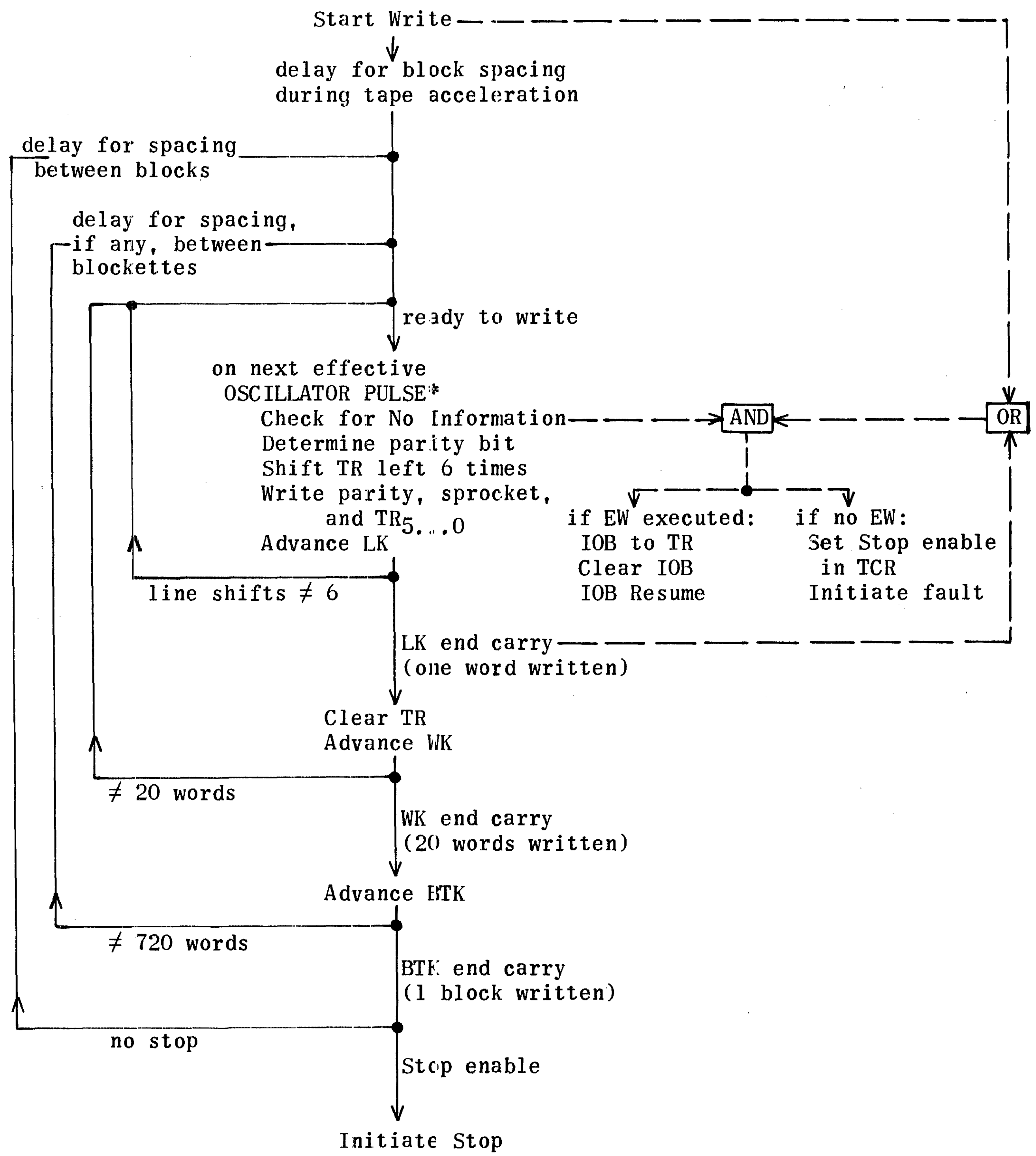
Tape Shift Counter (TSK) - a counter which regulates the shifting in TR of the six data <u>bits</u> of a line. On the completion of the shift of six bits (one line), a TSK "end carry" is propagated which is interpreted as an "advance LK" signal by the Line Counter; TSK is then cleared.

Line Counter (LK) - a counter which controls during reading and writing the shifting in TR of the six <u>lines</u> of a word. In a reading operation, when six lines have been assembled in TR, LK propagates a signal to effect TR to IOB; an "advance WK" signal is sent to the Word Counter, and TR and LK are "cleared". In a writing operation when six lines have been written on tape from $TR_{5...0}$, LK propagates a signal which clears TR and enables the transmission IOB to TR (if IOB has been filled by an EW instruction); an "advance WK" signal is sent to the Word Counter, and LK is "cleared".

In a moving operation LK is advanced by the receipt of a delayed sprocket pulse. When six lines have been "moved", an "advance WK" signal is sent to the Word Counter and LK is cleared.

Word Counter (WK) - counts the number of words read, written, or moved until a blockette count is reached; i.e., when a 20 word count is reached, an "advance BTK" is sent to the Blockette Counter and WK is "cleared".

Blockette Counter (BTK) - counts the number of blockettes read, written, or moved until a block count is reached; i.e., when a six blockette count is reached, a "BTK end carry" is sent to various portions of tape control, and BTK is "cleared". A BTK end carry is used during a moving operation to subtract "1" from the Block Counter.


Start Write
delay for block spacing during tape acceleration
delay for spacing between blocks
delay for spacing, if any, between blockettes
ready to write
on next effective OSCILLATOR PULSE*
Check for No Information
Determine parity bit
Shift TR left 6 times
Write parity, sprocket, and $TR_{5...0}$
Advance LK
AND
OR
if EW executed: IOB to TR Clear IOB IOB Resume
if no EW: Set Stop enable in TCR Initiate fault
line shifts ≠ 6
LK end carry (one word written)
Clear TR
Advance WK
≠ 20 words
WK end carry (20 words written)
Advance BTK
≠ 720 words
BTK end carry (1 block written)
no stop
Stop enable
Initiate Stop

* Oscillator pulses do not effect any writing initiation during any of the delays quoted and during the time the tape is moving through a bad spot. During a stop initiation, oscillator pulses are made temporarily ineffectual by virtue of the delay (for the inter-block space) caused by a BTK end carry. During this delay, a Master Clear from Stop Control renders them ineffectual until the next Write Sequence.

Figure 9. Write Sequence

PX 38

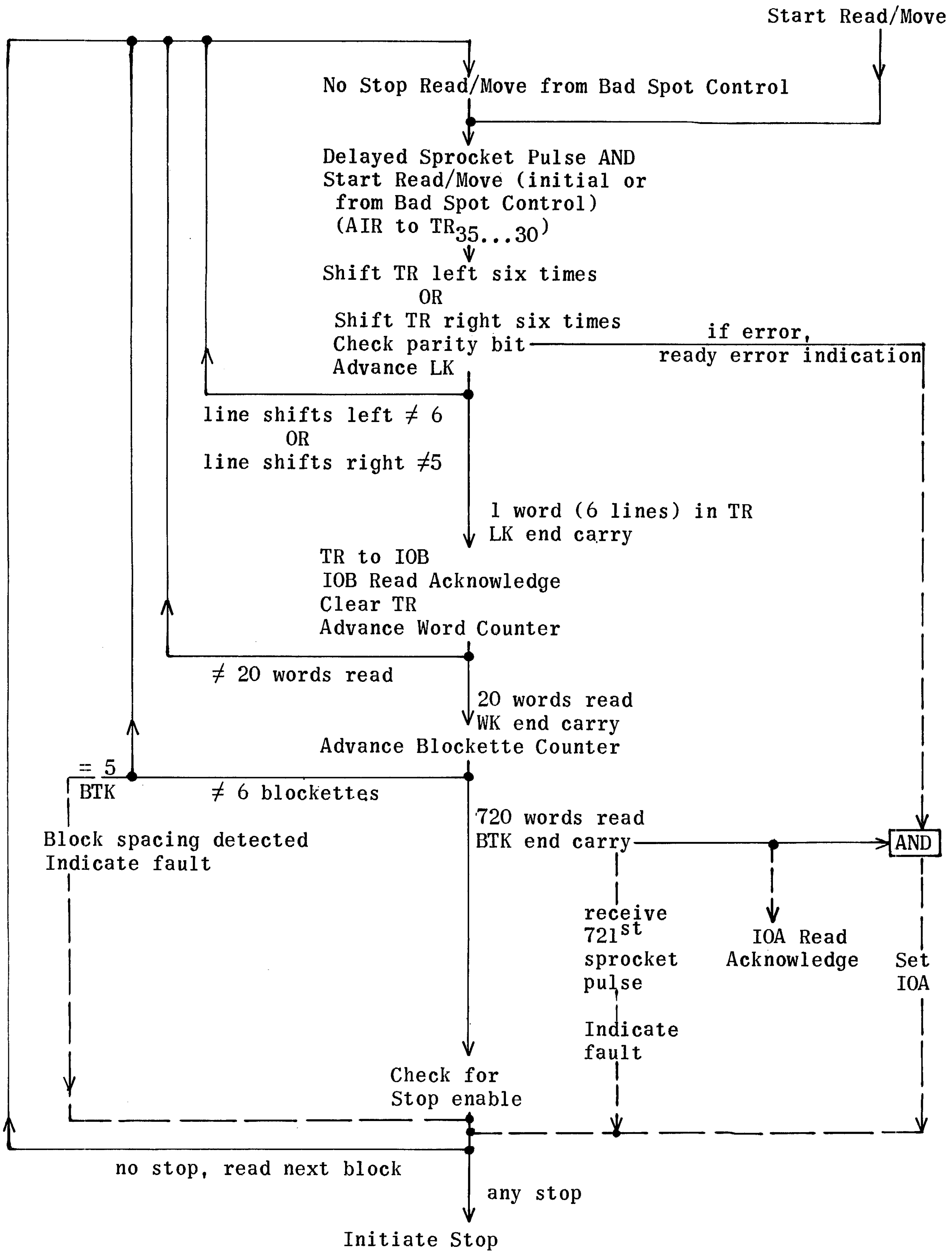


Figure 10. Read Forward Backward Sequence
(Delayed sprocket pulse approximately 40 μs after tape line to AIR)

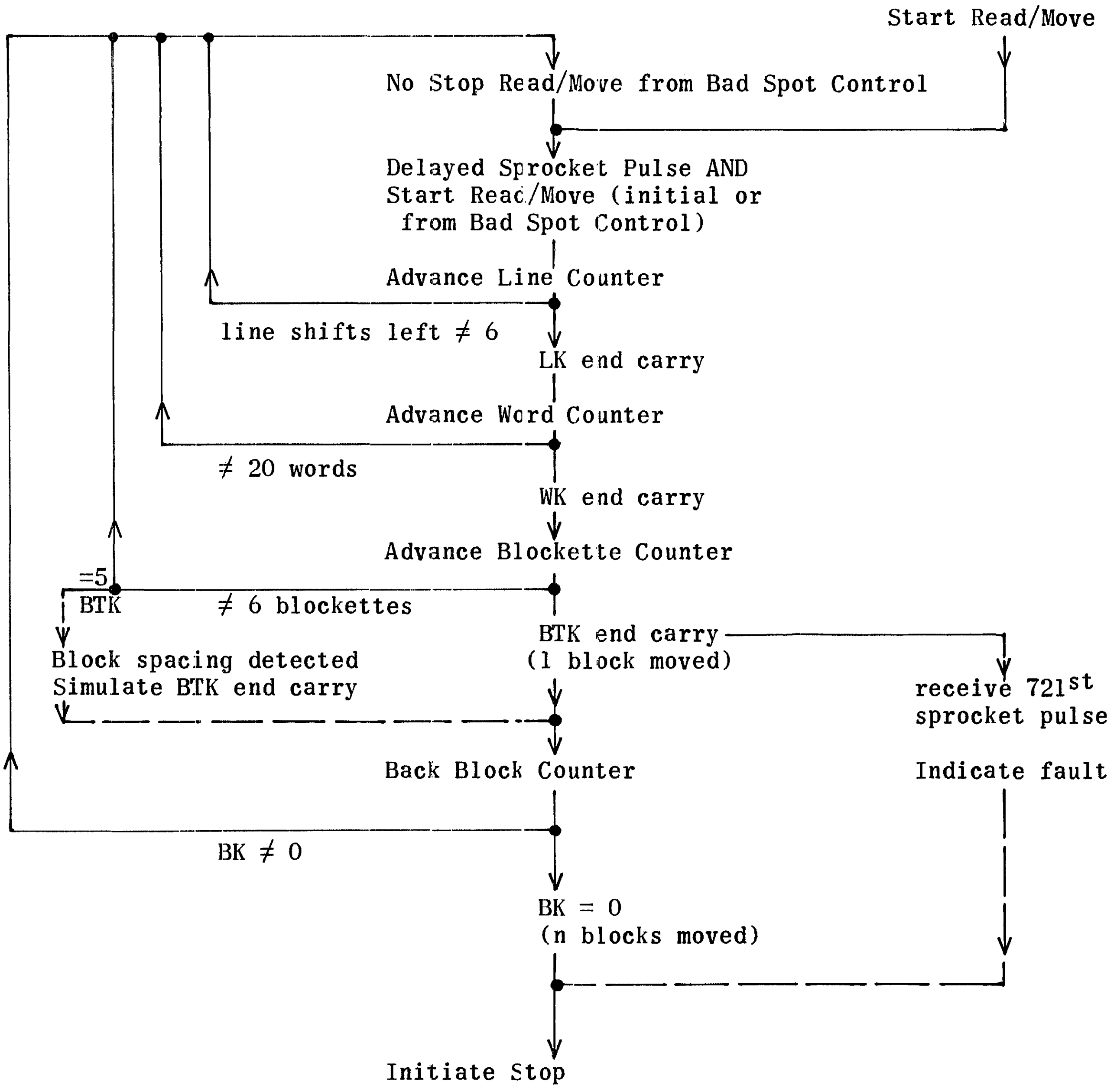


Figure 11. Move Forward or Backward Sequence
(Delayed sprocket pulse approximately 40 μs after tape line to AIR)

3 A BTK end carry <u>and</u> a Stop enable initiates a tape stop during which TCR is cleared.

4 A stop initiation prevents another IOB to TCR transmission for approximately 10 ms.

[Note that an IOB to TCR transmission can occur immediately after the last BTK end carry (unless a Read or Write One Block and Stop operation was programmed). Since TCR is not yet cleared, no operation except a Stop should be programmed immediately after reading or writing n (> 1) blocks.]

<u>Move Uniservo</u> j <u>n</u> <u>blocks</u>, n > 0

1 When and if j is available, the TCR lockout is set and the IOB Resume is given.

2 BTK end carry clears the TCR lockout (TSK end carry resets it) but IOB to TCR transmission is not allowed until TCR is cleared, removing the Move enable.

3 A BTK end carry from the n<u>th</u> block and a Move enable from TCR initiates a tape stop during which TCR is cleared.

<u>Stop Reading or Writing of n</u> (> 1) <u>blocks</u>

(Uniservo designation and read or write enable remain in TCR from the read or write operation to be terminated.)

1 Since j is available, the TCR lockout is set and the IOB Resume given.

2 The Stop enable now in TCR with the previous BTK end carry initiates a tape stop during which TCR is cleared and TCR lockout cleared. However, the stop initiation prevents another IOB to TCR transmission for approximately 10 ms.

<u>Rewind</u> j or <u>Rewind</u> j <u>with interlock</u>

1 When and if j is available, TCR lockout is set and IOB Resume is given.

2 When Rewind initiation is completed, TCR lockout is cleared and TCR is cleared.

<u>Bias Change</u>

1 TCR lockout is <u>not</u> set

2 Bias Change completion after approximately 20 ms effects IOB Resume and Clear TCR.

c. TAPE OPERATION TIMING.

(1) GENERAL. - Before a study is undertaken of the available computation times during tape operations, it is helpful to review the conditions which are necessary during tape operations before

(a) An External Function, External Write, or External Read instruction can be executed following an EF tape instruction and

(b) the transmission IOB to TCR can occur after an EF tape instruction is executed. (IOB is automatically cleared immediately after this transmission.)

The transmission X to IOB during the execution of an EF or EW instruction is not possible after the execution of a previous EF instruction until an IOB Resume is received by IOB Control. (A lockout condition is established by IOB when a <u>second</u> X to IOB transmission is attempted before an IOB Resume is received from external equipment. This prevents the use of IOB for a second output operation before the first is completed.) No wait for an IOB Resume is necessary when an ER instruction follows an EF instruction since the attempted execution of an ER instruction sets up an IOB lockout until information is received from external equipment (and this transmission is not received until a tape operation is underway). An ICB Resume generated by tape control after an EF instruction indicates that tape control has accepted the current content of TCR as a tape operation. Since this IOB Resume allows the loading of IOB by a second EF instruction or an EW instruction, TCR must be protected against receiving another IOB to TCR transmission until the first tape operation is completed. This protection is provided by setting the "TCR Lockout" before giving the IOB Resume. Then, since these transmissions are blocked, IOB may be loaded safely by another EF or an EW (or an ER). Since a Bias Change operation does not effect a TCR lockout, it does not generate an IOB Resume until the change of bias is completed.

After an EF tape instruction has been executed, the TCR lockout set up by the previous tape operation must be removed before the operation code currently in IOB can be transmitted to TCR. (In some cases, the removal of the TCR lockout is not sufficient to allow the IOB to TCR transmission.)

It should be noted that if TCR has not been cleared previous to an IOB to TCR transmission, the logical sum of IOB and the current content of TCR are formed in TCR. This is not allowable except when an EF stop instruction is programmed to terminate a Read or Write operation.

The occurrence of these signals, IOB Resume, set TCR lockout, clear TCR lockout, and clear TCR, during each of the legitimate tape operations is pointed out below.

<u>Read</u> <u>or</u> <u>Write</u> <u>Uniservo</u> j (<u>and</u> <u>stop</u>)

<u>1</u> When and if j is available, the TCR lockout is set and the IOB Resume is given.

<u>2</u> A BTK end carry clears the TCR lockout. (TSK end carry, if reading or writing is continued, resets it.)

(2) AVAILABLE COMPUTATION TIMES.

(a) A tape operation becomes effective when the operation specification, as placed in IOB by an External Function instruction, is transmitted to TCR. This transmission may not occur immediately after the execution of an EF instruction. The initiation of a tape stop on any Uniservo causes a Stop Initiation delay which prevents for 10 ms the following: IOB to TCR transmission, IOB to BK transmission, and Clear IOB signal. Consequently, the emission of the IOB Resume signal is also detailed until this time, and lonqer if a Change Bias operation is being initiated. During this 10 ms, an External Function instruction may be executed but the tape operation is not initiated. The attempt to execute an External Read or External Write instruction, if a read or write operation is to be initiated, or a second EF instruction, establishes an IOB lockout condition until the IOB Resume is emitted.

(b) Between an External Function instruction for a read or write operation on Uniservo j and the first External Read or External Write. It is assumed that Uniservo j is immediately available.

	Writinq operation	Readinq Operation
Possible Stop Initiation delay*	10 ms (maximum)	10 ms (maximum)

The following delays are incurred after a read or write operation is initiated, but before a word is transmitted to IOB (in reading) or (in writing) before tape control assumes a word has been received from IOB.

Tape Direction delays		
normal	2.5 ms	2.5 ms
if movement is to be in opposite direction from previous movement	600 ms	600 ms
Leader delays		
if tape is on leader	1500 ms	1000 ms
Block Spacing Delays		
for one inch block spacing	7.5 ms	5.0 ms **
for 2.4 inch block spacing	14.5 ms	5.0 ms **

The delays listed are "progressive", i.e., they are not initiated simultaneously. The delays listed detain signals internal to the tape control system with the exception of the block spacing delays listed for a reading operation.

* If the EF Read or Write instruction immediately follows an EF Stop instruction or the last ER or EW of a previous Read or Write One Block and Stop operation. See paragraph (a).

** Actual reading of the tape does not begin until the tape is moving through the read/write head at its free-running speed. The times quoted above are acceleration times for the tape to reach this rate. These delays assume no variation from a tape speed of 100 inches per second and are minimum times for this speed. The times of 7.5 or 14.5 are allowable assuming that block spacing is exactly 1.0 or 2.4 inches and that tape movement is stopped exactly in the middle of the block spacing. Only when the block spacing is known to be consistently 2.4 inches can the 14.5 ms time be used.

(c) Between successive External Read instructions or successive External Write instructions:

at a recording density of 128 lines per inch	436 microseconds
at a recording density of 50 lines per inch	1168 microseconds

Tape moves at a rate of 100 inches per second. Therefore, for a recording density of 128 lines per inch, 36-bit words are transferred to IOB (or sensed from IOB) at the rate of one word every 468 microseconds ($10^4 \cdot 6/128$ microseconds). When the execution time for the External Read (or External Write) instruction is subtracted from this time, 436 microseconds remain for computation. The available computation times for other recording densities are similarly computed.

(d) During the time a block or blockette spacing is moving past the read/write head:

	Writing operation*	Reading Operation**
One inch inter-block or inter-blockette space	9.968 ms	7.468 ms
2.4 inch inter-block space	23.968 ms	17.968 ms
0.1 inch inter-blockette space	.968 ms	.718 ms

(e) A maximum computation time of 250 microseconds can be used for other than tape operations between the External Read or Write instruction which reads the last word in the last block and the External Function instruction which initiates a tape stop.

(f) Computation time available after a Move Forward or Move Backward operation becomes effective (on IOB to TCR, IOB to BK). Another EF tape instruction may be executed immediately but it will not become effective until after times listed below.

Tape Direction delays	minimum 2.5 ms or maximum 600 ms if tape movement is to be in opposite direction from last movement.

* During a writing operation the writing pulses are cut off for 10,24, or 1 ms. These times minus the .032 ms execution time of the External Write instruction yield the times quoted above.

** Again the reading rate is dependent on the tape speed and the length of the inter-block and inter-blockette spaces. It is ample to allow for a 25% deviation in the length of spaces between blockettes and between blocks. Thus, the minimum available computation time, assuming tape motion at the rate of 100 inches per second and a space of 0.75 inches in length, is computed to be (7500 -. 32) microseconds while a "one inch" block space is moving by the read/write head.

Leader delay	1000 ms if tape positioned on leader
or	
Acceleration time*	5 ms if tape not positioned on leader

Free-running time* for n blocks:	
Block rate	
recording density 128 lines/inch	n x 56.25 ms
recording density 50 lines/inch	n x 144.00 ms
Inter-block space rate	
one inch inter-block space	(n-1) x 7.468 ms
2.4 inch inter-block space	(n-1) x 17.968 ms

Stop Initiation delay 10 ms

(g) Computation time available after a Rewind or Rewind with Interlock operation becomes effective (on IOB to TCR). Another EF tape instruction may be executed immediately but the IOB to TCR transmission for this operation is not allowed until after the times listed below:

Minimum 10 ms or

Maximum 600 ms if tape movement is to be in opposite direction from last movement

In addition to the times quoted above, if the tape instruction following a Rewind instruction references the same Uniservo, this next instruction will not become effective until the rewinding is completed. Maximum rewinding time is approximately three minutes. In this case, this time also could be used advantageously for other computations not referencing IOB.

(h) Computation time available after a Change Bias operation becomes effective. The attempt to execute immediately another External Function instruction establishes an IOB lockout condition. There is a 20 ms delay after the initiation of a bias change before another EF instruction can be executed to its completion.

d. OPERATION.

(1) OPERATION INDICATORS. - Tape operation is reflected by the condition of indicators on the Uniservos, the tape control cabinet, and the left section of the computer control panel. These indicators and their reactions to tape operation and tape operation faults are discussed in the following paragraphs.

(a) UNISERVO. - The "Ready" indicator (green) is between the tape reel panel doors, upper center section of the Uniservo. This indicator is illuminated when the Uniservo interlock circuit is energized; i.e., when power has been applied to the Uniservo, when the tape reel panel door switch is set to its ON position, and when the Forward Limit, Mylar Detector, Left Tape Loop, and Right Tape Loop switches are in their normally closed position. If this indicator is not illuminated, the Uniservo is not ready for operation.

* Assuming no variation from a tape speed of 100 inches per second.

A failure to have the interlock circuit not energized because power has not been applied to the Uniservos would not usually occur. Power is normally applied to the Uniservos (and the tape control system) at the same time power is turned on for the computer. Normally, if it is noted that the "Ready" indicator is not illuminated, the interlock circuit is not energized because one of the switches in the circuit is open. The condition which caused the switch to be opened must be corrected before any operation on the unit can be undertaken. If operation is attempted on a unit in which the interlock circuit is not energized, a computer B-Fault condition is incurred. The interlock switches and the conditions which cause them to open are discussed in the following paragraphs.

The tape reel panel door switch is located immediately below the "Ready" indicator. This interlock switch must be set to OFF to open the left tape reel panel door and cannot be returned to its ON position until the door is closed. Effectively, then, opening the Uniservo door causes the "Ready" indicator to be dropped. The door must be closed and the door switch reset to ON before the interlock circuit is energized.

A Mylar shim, or buffer tape, is inserted between the read-write head and the metallic recording tape. This plastic tape serves to reduce both tape wear and friction. When this tape is broken or the supply is exhausted, the Mylar detector interlock switch is opened, dropping the "Ready" indicator. The replacement of the Mylar tape by maintenance procedures closes this interlock switch.

The Forward Limit interlock switch is opened when the magnetic tape moves into its "leader" area on its far end, i.e., the left-hand tape reel is depleted and the right-hand tape reel contains all of the tape. (The Forward Limit switch is opened when the "rubber bumpers" on the "leader" on the far end of the tape are detected). To close this switch and energize the Uniservo interlock circuit, the tape must be rewound past its "leader" position. This is accomplished by opening the panel door and manually turning the reel in the counter-clockwise direction several times (until the switch no longer makes contact with the rubber bumpers). The complete rewinding of the tape onto the left-hand reel can then be accomplished by the normal rewind operation which can be instigated manually from the computer control panel or under program control.

The Right and Left Tape Loop switches are opened when the tape loops are out of normal position. This could be caused by tape breakage or possibly could result from faulty operation of control circuits in the Uniservo. Maintenance procedures are necessary to correct these conditions.

De-energizing the Uniservo interlock circuit could also be caused by blowing a fuse. A blown fuse in the Uniservo cabinet is shown by the illumination of the Fuse indicator.

Inside the right-hand tape reel door, above the tape reel mounting, are indicators labeled Rewind Interlock and Fault, Fuse and Temp.

The Fuse indicator is illuminated by a blown fuse. This indicates that some part of the Uniservo is inoperative and the Uniservo interlock circuit may be de-energized. Detection and replacement of the blown fuse are maintenance procedures.

The Temp. indicator is illuminated when a temperature rise above 120°F is detected in the Uniservo. This condition causes a computer A Fault and illuminates the Temp indicator in the A Fault Group on the computer control panel. Computer operation is halted by an A Fault condition. Operation is resumed after corrective maintenance by depressing the Clear A Fault button (unless a B-Fault has resulted from tape reading or writing occurring at the time of the computer stop. If this is the case, either (1), the MT fault indicator in the B-Fault group on the computer control panel and the No Information fault indicator in tape control cabinet are illuminated, or (2), the IO fault indicator in the B-Fault group on the computer control panel is illuminated).

The Rewind Interlock indicator is illuminated when the magnetic tape has been rewound with interlock on the left-hand tape reel. This condition indicates that the tape on this Uniservo should be replaced before this unit is used again. Opening the door to replace this tape drops the "Ready" condition of this unit (because the door switch must be set to OFF) and drops the rewound with interlock condition.

Inside the right-hand tape reel door, above the tape reel mounting, are indicators labeled Clutch, Stop (red) and Go (green). The Go indicator is illuminated by the signal which is sent to the tape drive mechanism to pull in the clutch and start tape movement. This indicator remains illuminated until a stop tape signal is received by the tape drive mechanism to operate the brake, thus releasing the clutch. At this time the Stop indicator is illuminated and remains illuminated until the tape is re-started or until power is dropped from the unit.

(b) TAPE CONTROL CABINET. - This cabinet is located immediately to the left of the Power Supply Cabinet. Located inside the right-hand door are tape fault indicators and the Logical Number Selection switches. Each Uniservo is physically defined by one of the numbers 1, 2, ... 10, depending upon the number of Uniservos installed. For instance, if an installation has eight Uniservos, it would be expected that the numbers 1 ... 8 would define the eight units. The Logical Number Selection switches are labeled Uniservo 1, Uniservo 2, etc. Encircling the switches are the numbers 1 through 10. A Uniservo is assigned a logical designation by turning the appropriate selection switch so that the white line on the switch is in line with the number desired. The numbers available for logical assignment depends upon the number of Uniservos installed, i.e., if eight Uniservos are installed, any of these units may be logically assigned any of the numbers one through eight. Thus, if the eight Uniservos at an installation are physically defined as Uniservos 1 ... 8, the switches labeled Uniservo 9 and Uniservo 10 should be set to the logical designations of 9 and 10. It is not allowable for two switches to be set to the same logical designation even though some of the switches define a non-existent Uniservo.

The tape fault indicators (red) above the Logical Number Selection switches are labeled. No Information, Sprocket Error, Selection Error, and Uniservo Interlock. These indicators are illuminated by the detection of one of these tape faults. These faults also cause the illumination of the MT indicator in the B-Fault group on the computer control panel and cause a computer B-Fault stop.

The logical number Selection Error is caused at any time the computer is in operation by setting two Logical Number Selection switches to the same number, i.e., giving two "Uniservos" the same logical designation. The fault occurs regardless of whether or not the "Uniservo" is non-existent or out of service for maintenance reasons. This fault must be corrected before computer operation can be resumed. Correcting the selection causes the Selection Error fault light to be extinguished. Operation is resumed by depressing the Master Clear Button (which drops the B-Fault light), making the desired selections on the computer control panel, and depressing the computer Start button.

In order to prevent a Selection Error fault from occurring at the time of making a logical number designation change, the computer operation must be stopped. Depressing the Force Stop button on the computer control panel will allow a setting to be made safely. Computation is re-started by depressing the Start Button.

The Uniservo Interlock Fault is caused by referencing for a tape operation a unit in which the interlock circuit is de-energized. This condition is shown on the unit referenced by the extinguishment of the "Ready" indicator. The causes of this condition have already been discussed in the paragraphs discussing the "Ready" indicator and Uniservo interlock circuit. Computer operation can be resumed by depressing the Master Clear button, which extinguishes both the MT and Uniservo Interlock indicators, making the desired selections on the control panel, and depressing the Start button; but, if the cause of the Uniservo Interlock fault is not corrected and the same unit is again referenced, the B-Fault will re-occur. Correcting the Uniservo Interlock fault during a B-Fault Stop extinguishes both the Uniservo Interlock indicator and the MT fault indicator. Operation is resumed by depressing the Master Clear button (which drops the B-Fault indicator), making the desired selections on the control panel, and depressing the Start button.

The No Information and Sprocket Error faults are discussed later in the fault section. These faults can be cleared, and their indicators, the MT Fault indicator, and the B-Fault indicator are extinguished, by depressing the Master Clear button on the computer.

When the occurrence of an MT B-Fault is noted, observing the condition of the tape fault indicators in the tape control cabinet is an aid to diagnosing the cause of the fault. Before the cabinet door is opened, the Bypass Cabinet Interlock key in the Test Switch group, right section of the computer control panel, must be turned to its Abnormal position. The failure to do this before opening a cabinet door causes an emergency power drop to the computer system, and maintenance procedures are necessary to resume operation.

(c) SUPERVISORY CONTROL PANEL. - Represented on the left section of the Supervisory Control Panel are components of the tape control system. The banks of lights which assist the operator in manual operation of the Uniservos and aid in diagnosing certain of the fault conditions and computer operation "delays" are those labeled TCR, Tape Control Register, TR, Tape Register, and BK, Block Counter. The button labeled MT Test Start Step in the MT Test Writing Rate group is depressed to manually initiate a tape operation. The illumination of one of the indicators labeled MT Read Bias, high or low, shows the selection of a read bias other than normal.

The bank of lights labeled Center Drive Control, Start and Stop, indicate tape movement on the Uniservos. An indicator in the top row is illuminated by an "operate clutch" signal to a particular unit; this indicator is extinguished by an "operate brake" signal which illuminates the indicator in the bottom row. The numbers below the ten pairs of indicators, between the "set" buttons (black) and "clear" buttons (white) are the physical definitions of the Uniservos. If the need for an immediate stop of tape movement should arise, this can be effected, <u>if</u> the computer is not in operation at the time or if operation is in the Test mode, by depressing the "clear" button for the appropriate Uniservo. (Depressing the computer Force Stop button stops operation.)

(2) PREPARATION FOR OPERATION. - The procedure for preparing for tape operation under program control or manual control is as follows (assuming that the Uniservos have been properly fitted with tape reels):

(a) Determine whether those Logical Number Selection switches in the tape control cabinet which physically define installed Uniservos have been set to the logical number designations used in the program.

NOTE

(The numbers which can be used for logical designations cannot exceed the number of installed Uniservos.) If any logical designations are to be changed and the computer is in operation, the Force Stop button must be depressed before making the switch changes to prevent the occurrence of a Selection Error.

(b) Check for the illumination of the green "Ready" indicators on the Uniservos to be used. If this indicator is not illuminated, the "not ready" condition must be corrected before this unit can be used. Assuming that power has been applied to the Uniservos, the attempt to use any unit not ready causes a Uniservo Interlock B-Fault.

(c) Check for the illumination of the Rewound Interlock indicators on the Uniservos to be used. This condition should be eliminated before attempting any operation on the Uniservo. A tape reference to a Uniservo which has a Rewound Interlock condition causes a computer stall, and until this tape reference is removed from tape control the attempt to eliminate the Rewound Interlock condition causes a Uniservo Interlock B-Fault.

The procedure to replace any rewound tape is as follows:

(a) Turn the door switch to its OFF position and open the left tape reel panel door.

(b) The small spring clip connecting the magnetic tape and the leader should be positioned immediately below the tape reel. This connection is broken by spreading the sides of the spring clip, thereby releasing its prongs from the small cylindrical ending of the leader.

(c) Pull forward the holding latch on the tape reel mounting. This releases a locking pin directly under the knob from its position in one of the slots in the inner circumference of the tape reel. Remove the tape reel from the tape reel mounting.

(d) Place another rewound tape on the tape reel mounting so that the tape winding is in the clockwise direction. Return the holding latch to its closed position, first positioning the tape reel so that the locking pin is inserted into any of the slots in the inner circumference of the tape reel.

(e) Connect the magnetic tape to its leader by inserting the prongs of the spring clip in the cylindrical ending of the leader.

(f) Turn the tape reel counterclockwise until any tape slack is taken up.

(g) Close the panel door and <u>set the door switch to its ON position</u>.

(3) MANUAL OPERATION. - To initiate tape operations from the computer control panel, the computer <u>must</u> be set to operate in the Test mode. The operations which can be successfully completed after a manual initiation are Rewind, Rewind Interlock, Move Forward and Backward, and Change Bias. To initiate one of these operations, set the desired operation code <u>and</u> Uniservo selection (except for a Change Bias operation) in TCR by depressing the appropriate "set" (black) buttons. (The white button is depressed to clear the register.) Manual settings may be made any time the computer is not in operation and when operation is in the Test mode. If a Move operation is desired, the number n of blocks to be moved in inserted in BK. Depressing the MT Test Start Step button then causes the specified tape operation.

At the completion of the Move Forward, Move Backward, and Change Bias operations, the Tape Control Register and the Block Counter are automatically cleared. If the bias is changed to high or low, one of the bias indicators (red) in the MT Test Writing Rate group is illuminated. If a manual rewind operation is performed, TCR is cleared when the rewind operation is initiated.

The failure of the registers to be cleared as expected after attempting a manual operation could be due to one of the following reasons: (1) The Test mode of operation was not selected; (2) for a rewind or move operation, no Uniservo selection was made in TCR or the Uniservo specified is not available; (3) for a move operation, the block count set in TCR was greater than the nmmber of blocks available for moving on the tape. This is evidenced by a reduced block count $\neq 0$ in BK and a Uniservo Interlock B-Fault if the move was in a forward direction.

If a manual Move Forward or Move Backward operation is attempted without inserting a block count in BK, TCR is cleared and no tape movement is initiated.

e. IMPROPER PROGRAMMING OR OPERATION.

(1) GENERAL. - Consideration must be given to the effects of improper programming and operation errors on both tape operation and computer operation. In general, a program should run correctly if the equipment has been prepared properly, the Uniservo designated is available, a legitimate tape operation has been specified, the correct number of External Reads and External Writes have been coded, a Stop has been coded to terminate a Read or Write operation, and the timing restrictions have been noted in coding, in particular, the External Read instructions, External Write instructions, and an EF Stop instruction. A disregard for these requirements may result in an operation "delay" (a temporary halt of operation) which (1) may not be immediately noticed or (2) may cause indirectly a computer fault; or a disregard for these requirements may cause directly a computer fault. Any time computer operation is stopped, tape operation should be stopped to prevent the possibility of a "runaway" tape, i.e., a tape which moves free of control to its "leader" position on the far end.

If erroneous operation is detected by the tape control system after tape movement has been initiated, the tape control system effects the tape stop and initiates a B-Fault computer stop. The tape stop occurs at the end of the block (i.e., midway of the next interblock space) which was being read, written, or moved at the time of the fault detection. At the time the "stop tape" signal is sent to the Uniservo, the Tape Register, Tape Control Register and tape counters are cleared. In some cases, the execution of computer instructions may continue until the time of the actual computer stop, approximately 80 ms after the detection of the tape fault. If this is the case, and an External Function instruction for a tape operation is executed during this time, the possiblity exists of starting another tape movement. If this next EF instruction executed is an EF Stop (Read or Write) instruction, the lack of a Uniservo designation in TCR prevents the initiation of tape movement and effects an IOB lockout condition. For this reason, it is important that an EF stop instruction does not specify any particular Uniservo. If tape movement is started erroneously, depressing the computer Master Clear button stops the tape movement. (To stop tape movement without clearing any of the registers on the computer control panel, a button in the Center Drive Control can be depressed as described in subparagraph entitled Supervisory Control Panel, Operation.)

When computer operation is stopped by a computer A-Fault, a B-Fault, a Force Stop, or a Manually Selective programmed stop, any tape movement in progress at the time of the actual operation stop is halted (1) during a rewind operation, at the completion of the rewind; (2) during a move operation, at the completion of the move; (3) during a read or write operation at the end of the current block being read or written. This block is not read completely or written correctly to completion since the appropriate number of External Reads and External Writes are not executed.

In diagnosing the cause of an operation delay or computer fault, noting the condition of the following indicators is helpful.

(a) On the Supervisory Control Panel:

The registers in Magnetic Tape Control, in particular, TR, TCR, BK.

The IOB Register, located above TR.

The Program Control Register (PCR) and Program Address Counter (PAK) both located in the center section of the control panel. At any time,PCR holds the instruction being executed and PAK holds the address of the next instruction to be executed.

The IO and MT fault indicators in the B Fault group, lower center section of the control panel. The illumination of the IO fault indicator indicates the occurrence of one of the IOB (or IOA) external faults shown in the External Fault group, lower left section of the control panel. The illumination of the MT fault indicator shows the occurrence of some other fault originating in the tape system.

The Temp fault indicator in the A-Fault group.

An IOB lockout condition is shown by (1) the illumination of the top-most light in the column labeled Wait External in the Pulse Distribution Control group, center section of the control panel, and (2) the presence of an ER,EW, or EF instruction in PCR.

(b) The indicators on the Uniservos: Ready, Rewind Interlock Fuse, and Temp.

(c) The Fault indicators in the tape control cabinet: No Information, Sprocket Error, Selection Error, and Uniservo Interlock.

Note: The door of the Tape Control Cabinet must not be opened without first turning the Bypass Cabinet Door Interlock key (in the Test Disconnect switches, right section of the computer control panel) to its Abnormal position.

(2) TAPE OPERATION FAULTS. - The operation errors which may occur during tape operation are listed below and discussed in detail subsequently.

Parity Check Error:	effects a stop of tape movement; caused by a reading or recording error.
Temperature Fault:	effects a computer A-Fault; equipment fault.
Uniservo Interlock:	effects a computer B-Fault; operations error or equipment fault.
No Information:	effects a computer B-Fault; programming error.
IO Read Fault:	computer B-Fault; programming error.
Selection Error;	effects computer B-Fault; operator's error
Sprocket Error:	effects computer B-Fault; recording error

The detection of a parity check error does not effect a computer fault but does effect a stop of tape movement. At the completion of reading every block, a check is made to determine if a parity error occurred in reading any of the 720 lines in the block. If one or more errors occurred, a "1" is set into stage 0 of IOA and a tape stop is initiated. If no error was detected, only the IOA Read Acknowledge signal is sent to IOA. Thus, IOA must be "read" and its content tested after reading every block. Computation continues depending upon the result of the test. If a parity check error is indicated, the block can be re-read in the opposite direction, and read and re-read at the different bias levels. If none of these passes effect a correct reading, a computer stop can be programmed to indicate the unsuccessful attempt to read the block correctly.

(a) UNISERVO TEMPERATURE FAULT.

Indications: Illumination of the Temp A Fault indicator on the computer control panel. Illumination of the Temp Fault indicator on a Uniservo.

Diagnosis: The temperature fault results from a temperature rise above 120°F in any Uniservo. The occurrence of a Temp Fault during tape operation does not interfere with the tape operation unless reading or writing is occurring at the time of the computer stop. If this is the case, a computer B Fault is also incurred. This is evidenced by the illumination of the MT Fault indicator on the computer control panel and the No Information fault indicator in the Tape Control Cabinet, or the illumination of the IOB Fault indicator on the computer control panel.

Resumption of Operation: If a B Fault has not occurred, turning the Bypass Temperature Interlock key to its Abnormal position after the computer stop has occurred allows resumption of the program in the <u>Test</u> mode after the Clear A Fault button is depressed. However, this procedure should be undertaken with caution. To resume computation in the Normal mode, the Temp fault must be corrected. The correction of the fault extinguishes the Temp Fault indicator on the Uniservo. Operation is then resumed by depressing the Clear A Fault button which extinguishes the Temp Fault indicator on the computer control panel.

If a B Fault condition exists, the A Fault is cleared by one of the procedures above; clearing the B Fault condition requires the depression of the Master Clear button. Operation is resumed according to the type of read or write fault incurred. These faults are discussed subsequently.

(b) UNISERVO INTERLOCK FAULT

Indications: Illumination of MT fault indicator and B Fault indicator in B Fault group, computer control panel. Illumination of Uniservo Interlock fault indicator in tape control cabinet. The "Ready" light on the Uniservo in use at the time is extinguished.

Diagnosis: This fault is caused when a Uniservo interlock circuit is de-energized and this particular Uniservo is referenced for tape operation. The conditions which cause the Uniservo interlock circuit to be de-energized and the correction of these conditions are discussed in the

Operation section in the paragraphs describing the "Ready" indicator on Uniservo and the Uniservo interlock circuit. If the not ready condition is detected by the attempt to initiate a tape operation, tape movement is not started. The execution of computer instructions may continue until the time of the computer fault stop. During this time, if External Write instructions are executed, "writing" is performed on the stationary tape; if External Read instructions are programmed to be executed, an IOB lockout condition is established.

If the Uniservo interlock circuit is de-energized after a tape operation and tape movement have been initiated, the execution of instructions continues until the time of the computer fault stop. Tape movement is stopped by virtue of the drop of power on the circuit.

Resumption of Operation: Depressing the computer Master Clear button extinguishes the fault indicators and allows the resumption of operation. Correction of the fault condition is not necessary to resume operation, but if the same Uniservo is referenced again, the fault will re-occur.

(c) NO INFORMATION FAULT: The failure to have a word transferred from the computer to TR results in a NO INFORMATION FAULT.

Indications: On the computer control panel MT B fault indicator is illuminated. In the tape control cabinet, No Information fault indicator is illuminated. TCR is cleared by the fault stop but may be filled by the execution of another EF tape instruction before the computer is stopped. Tape movement is stopped at the end of the block being written when the error occurs. The Computer is stopped approximately 80 ms after the completion of writing the block in which the fault was detected.

Diagnosis: This fault is caused by the failure to provide 120 External Writes to write a block or by the failure to execute in time an External Write. When this fault is detected by tape control, the stop code bits are set in TCR and a B fault is initiated.

If the fault is caused by programming too few External Write instructions, the bits of the missing words are written as zeros.

If the No Information fault is caused by the failure to execute an External Write in time, the data bits of the lines written on tape at that particular time will be zeros until the EW is executed; then, the transmission IOB to TR occurs and the remainder of the lines written is taken from the current content of TR.

If the block being written when the fault is incurred is not the last block programmed to be written, the attempt to execute the next group of External Writes will cause an IOB lockout condition. If this block was the last block to be written in this particular writing operation, the execution of computer instructions, including any External Function instruction, may continue until the time of the computer fault stop.

Resumption of Operation: All fault indicators are extinguished by depressing the Master Clear button. After remaking selections, operation can be resumed by depressing the Start button.

(d) IO READ (IOB I) FAULT, i.e., Failure to execute a sufficient number of External Reads: where 120 n-j (j=1...), ER's programmed and an EF stop programmed to terminate presumably the reading of the nth block: Failure to execute an External Read in time.

Indications: Illumination of the IOB fault indicator and the IOB I fault indicator. Tape movement is stopped at the end of the block being read when the error occurs. TCR is cleared by the fault stop but may be filled by the execution of another EF tape instruction before the computer is stopped. The computer is stopped approximately 80 ms after the fault is detected.

Diagnosis: Each word received by IOB from the Tape Register should be removed from IOB by an External Read instruction before the next transmission from TR occurs. If a second transmission occurs before IOB is cleared by an ER, an IOB I computer B Fault is incurred. In a tape reading operation this fault is caused by the failure to execute the sufficient number of ER's, or it could occur when an ER is programmed to be executed too late.

If two or more External Reads are not programmed for reading a single block or reading the last of a series of blocks, the tape is stopped at the end of this block but computer instructions continue to be executed. The execution of an External Function instruction during this time will transmit tape operation codes to IOB which has not been cleared. Thus, IOB will contain the logical sum of its previous contents and the tape operation code, and these bits are transmitted to TCR. No prediction can be made as to whether or not a tape operation will be initiated since both the operation code and the Uniservo selection may have been changed.

If two or more External Reads are not programmed for reading any but the last of a series of blocks, tape movement is stopped at the end of the block and the execution of the second ER for the next block sets up an IOB lockout condition which stops the execution of further instructions until the time of the computer stop. (The execution of the first ER for the next block transmits to storage from IOB the logical sum of the last two words of the last block.)

The lack (in the program) of one External Read in reading a single block or the last of a series of blocks does not cause directly a computer fault; but since IOB is not cleared before another EF instruction is executed at any future time, the bits transferred from IOB to TCR at that time may not specify the desired tape operation. Again, a tape operation may or may not be initiated. If an EF Stop Read instruction is executed, the logical sum of IOB and the specification of the read operation (currently in TCR) is formed in TCR. If the Uniservo selection is changed, tape movement on this Uniservo is <u>not</u> stopped and tape movement on another Uniservo may be started.

The lack (in the program) of an External Read in reading any particular block but the last could cause an IOB I fault at the beginning of the next block or could be interpreted as a missing ER for the last block of the series.

If an External Read is not executed in time during the reading of any block, tape movement is stopped at the end of this block and an IOB lockout condition is caused by the execution of the "extra" ER at the end of this block. This stops the execution of any further computer instructions until the computer fault stop.

Resumption of Operation: All fault indicators are extinguished by depressing the Master Clear button. Operation is resumed after remaking basic selections by depressing the Start button.

(e) SELECTION ERROR FAULT.

Indications: Illumination of the MT and B fault indicators on the computer control panel and the Selection Error indicator in the tape control cabinet.

Diagnosis: This fault is caused by setting two of the Logical Number Selection switches to the same number at any time the computer is in actual operation. Any changes in logical designations should be made during a computer stop.

Resumption of Operation: Operation cannot be resumed until the logical designations are corrected. When this is done, the Selection Error indicator is extinguished. Depressing the Master Clear button extinguishes the B fault indicator and allows resumption of operation.

(f) SPROCKET ERROR FAULT: In reading, >720 or <720 line count; in moving, >720 line count.

Indications: MT B Fault indicator illuminated. Sprocket Error indicator in tape control cabinet illuminated.

Tape movement is stopped at the end of the block recorded with the improper number of lines. A computer fault stop occurs approximately 80 ms after the detection of the fault.

Diagnosis: A sprocket error results from (1) the detection of a block spacing before a count of 720 lines has been accumulated in reading or moving, and (2) the reception of a sprocket signal from the tape <u>after</u> a 720 line count has been accumulated in reading or moving and <u>before</u> the block spacing is detected. The detection of a <720 line count during moving is ignored: this condition propagates a "false" 720 line count to the tape control system.

During a reading operation, a <720 line count means that an "extra" External Read is executed at the end of the block. The execution of this ER establishes an IOB lockout condition which prevents the execution of further instructions until the computer is stopped. This lockout condition is indicated by an ER instruction in PCR.

During a reading operation, if a >720 line count occurs in a block which is not the last of a group of blocks being read, the execution of an External Read after tape movement is stopped establishes an IOB lockout condition and prevents the execution of further instructions until the computer stop. If a >720 line count occurs in the last block being read, another External Function instruction may be executed before the computer stop occurs, and another tape movement could be initiated. An additional word is sent to IOB for each extra <u>six</u> lines recorded on tape, if such should be the case. Then, since IOB is not cleared before the computer fault stop, if an EF instruction is executed, the logical sum of IOB and the tape operation specification is sent to TCR.

When a $>$ 720 line count is detected during moving, computer instructions may continue to be executed until the time of the computer stop.

Resumption of Operation: Depression of the Master Clear button extinguishes all fault indicators. Operation may be resumed after remaking basic selections by depressing the Start button.

OPERATING THE COMPUTER

1. GENERAL.

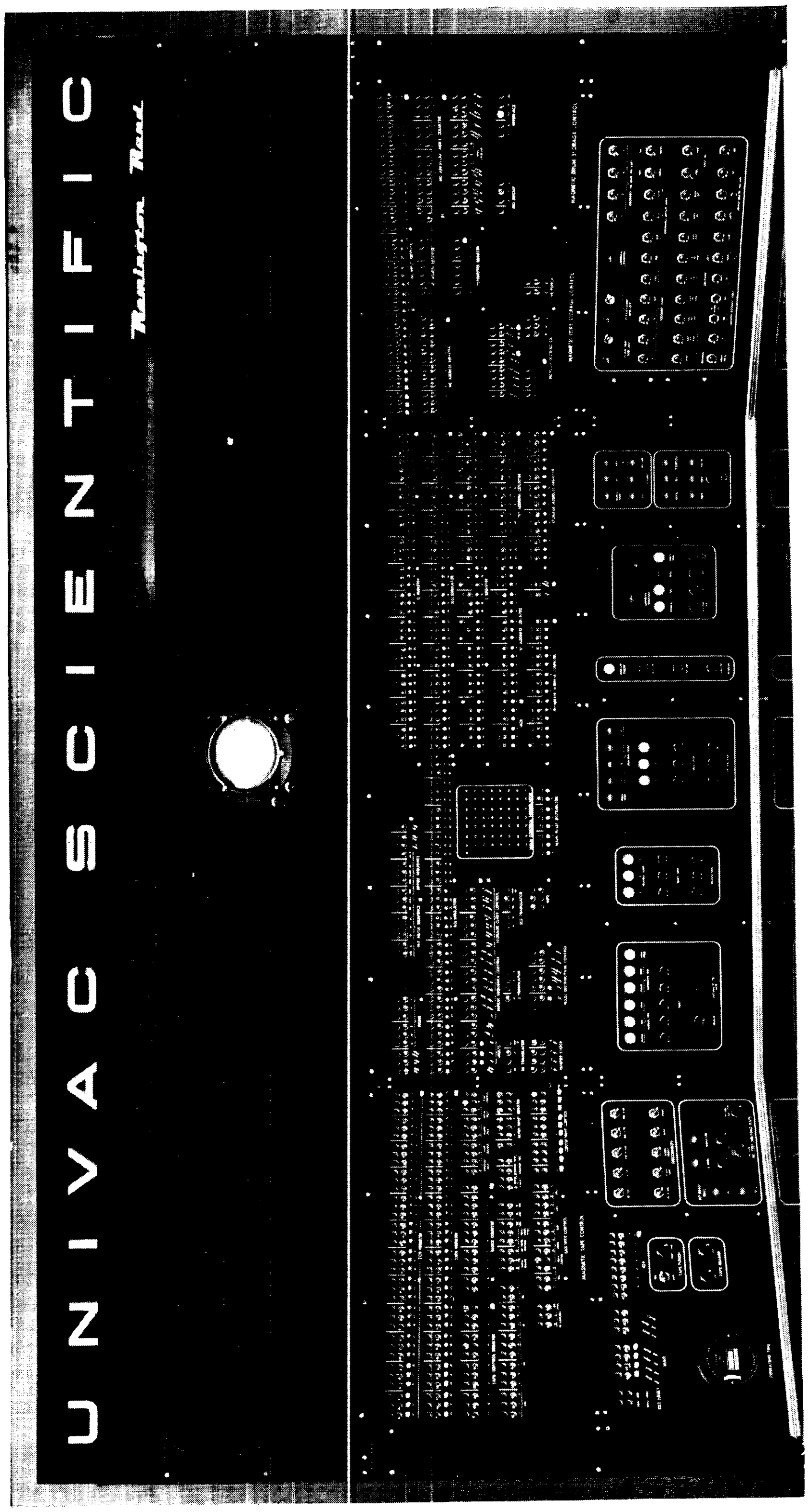
The Univac Scientific computer is set into operation by certain combinations of selections made on the Supervisory Control Panel. An overall view of the Supervisory Control Panel is presented in Figure 1. The computer may be set into high speed operation, or, if it is desired to manually superintend the internal actions of the computer and/or have a visual presentation of these internal actions, step operation may be chosen. Those internal operations which are represented on the Supervisory Control Panel, or which occur in components of the computer represented on the Supervisory Control Panel, have as their visual counterpart the occurrence of lights in the corresponding designated positions. Thus, the contents of any of the registers represented may be noted by interpreting the double rows of lights into a bioctal code as follows: a light in the upper row represents a binary one in the stage as numbered; and a light in the lower row represents a binary zero in that stage. Thus, each group of three columns of lights represents an octal number.

In addition to the ability to oversee the operations of the computer as it executes a program already internally stored, operations may be performed upon information placed in the computer by manually setting it in the counterparts of the proper registers on the Supervisory Control Panel. The small button at the lower right end of each register is depressed to clear the register (lighting the lower row indicators); the small button below each column of two lights is depressed to place a "1" in the chosen stage of the register (lighting the upper row indicator). By following the proper procedure, the contents of these registers may then be used as desired in computer operations.

Also, by following the proper procedure, instructions may be manually placed in the computer by inserting them in the Supervisory Control Panel counterpart of the PROGRAM CONTROL REGISTER (MCR, UAK, and VAK), and setting the MAIN PULSE DISTRIBUTOR to zero. If MPD is set at six, the first instruction to be executed is taken from the address shown in PAK (as automatically or manually set).

In discussing the operating selections which are made on the Supervisory Control Panel, the "group" designations listed below will be used. Each group is represented on the control panel by a set of pushbuttons, switches, and/or indicator lights enclosed in white lines. As they are located on the lower center section of the Supervisory Control Panel, Figure 2, from left to right, the groups are:

Operating Rate Group
Selective Jumps Group
Selective Stops Group
Program Interrupt Control Group
Operating Group
B Fault Group (upper)
A Fault Group (lower)



0 10416

Figure 1. Supervisory Control Panel, Overall View

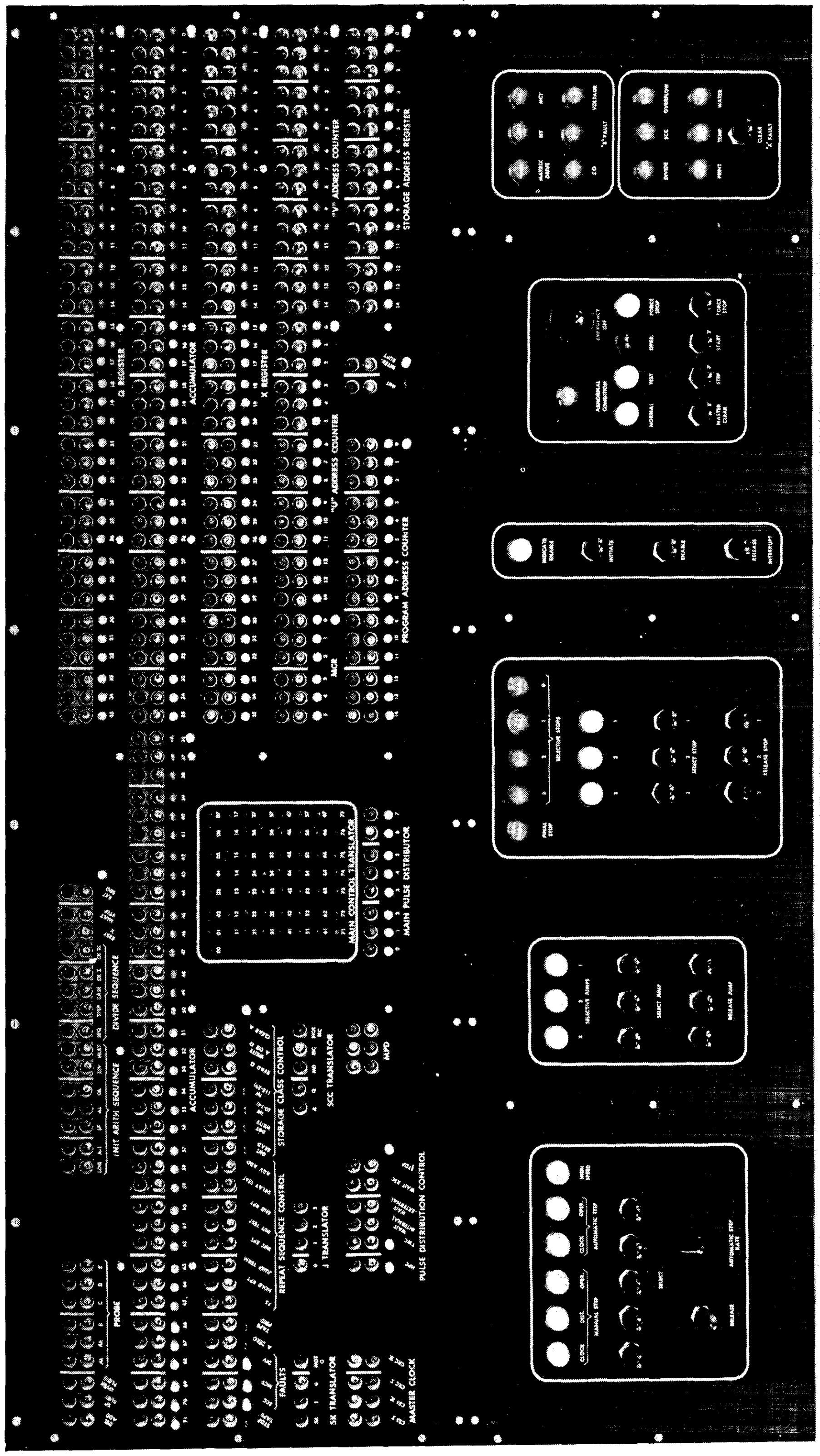


Figure 2. Supervisory Control Panel, Center Section

0 10415

As located on the right section of the Supervisory Control Panel, Figure 3, Test Switch Group.

As located on the left section of the Supervisory Control Panel, Figure 4, MT Disconnect Switch Group.

2. OPERATION.

a. GENERAL. - Computer operation is in one of two modes: NORMAL or TEST. For each mode, selections are made manually by depressing buttons and setting switches. The NORMAL mode is automatically selected unless the TEST/NORMAL switch (Test Switch Group) is set to TEST, or the MD NORMAL/ABNORMAL Switch (Test Switch Group) is set to ABNORMAL. Depression of any of the buttons in the Operating Rate Group also places the computer in the TEST mode. Operation at high speed is automatic in either the NORMAL or TEST modes. If high speed operation is not desired, a manual selection of step operation is necessary, i.e., one of the three buttons labeled MANUAL STEP, or either of the buttons labeled AUTOMATIC STEP (Operating Rate Group). (The depression of one of these buttons automatically selects the TEST mode of operation regardless of the TEST/NORMAL switch being set to NORMAL.) The AUTOMATIC STEP RATE switch controls the time rate at which Automatic Step Operation or Automatic Step Clock operations are performed. This timing control is applied to the rate at which instructions are executed if AUTOMATIC STEP OPERATION is selected. If AUTOMATIC STEP CLOCK is selected, the clock pulse rate is regulated accordingly. Timing during Manual Step operations is controlled by the selection of CLOCK, DISTRIBUTOR, or OPERATION, and the manual depression of the STEP button in the Operating Group. Each depression of the STEP button releases, respectively, one clock pulse, one distributor pulse, or the sequence of pulses necessary to the execution of one instruction. Depression of the RELEASE button in the Operating Rate Group releases any selection made and returns the computer to HIGH SPEED and to the NORMAL mode unless the TEST/NORMAL switch was actually set to TEST.

Operation in the NORMAL mode is not possible if any of the disconnect switches in the Test Switch Group, or MT Disconnect Switch Group, are positioned to their abnormal condition setting. This condition is indicated by the illumination of both the NORMAL indicator light and ABNORMAL condition light (Operating Group).

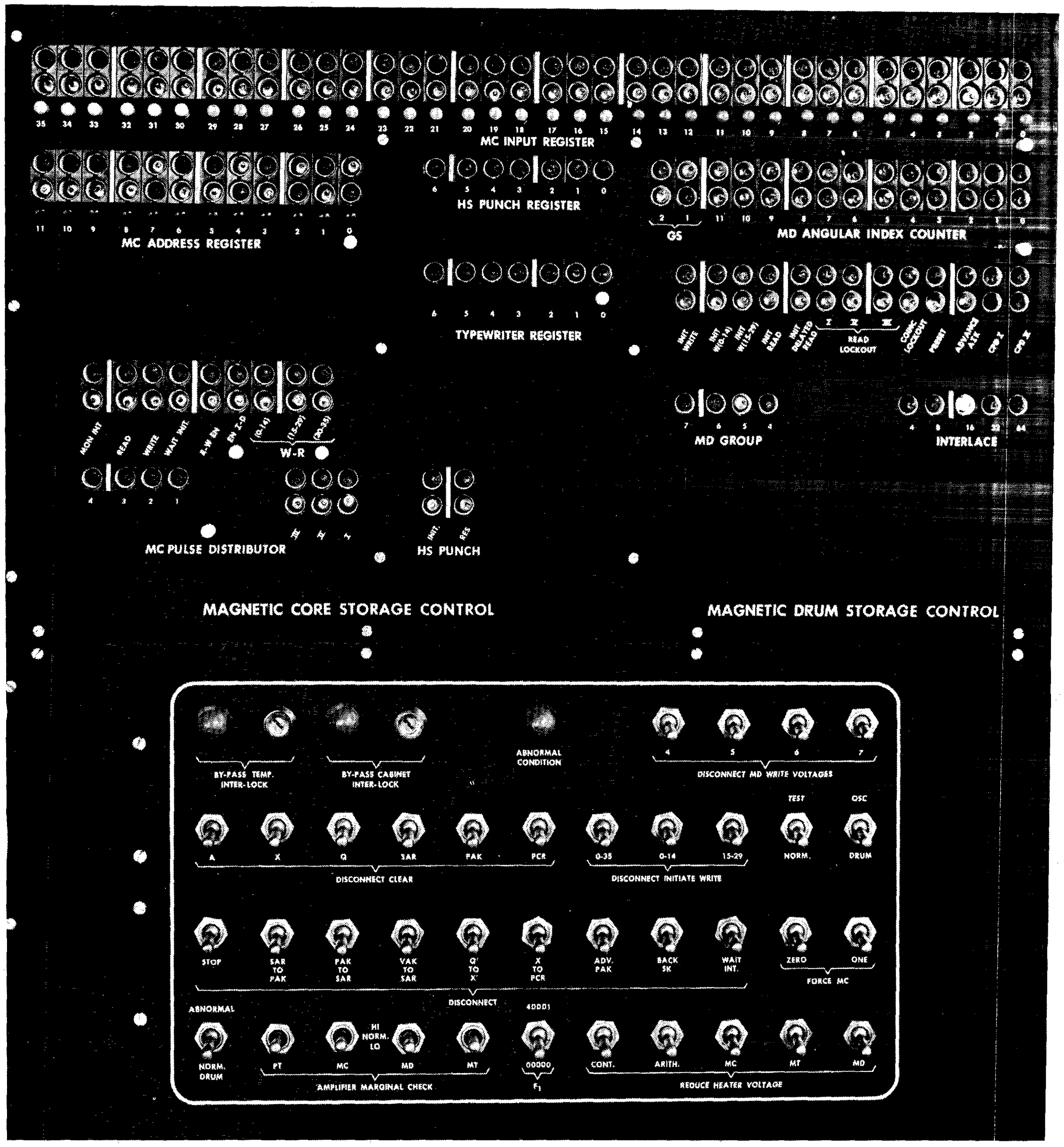
b. NORMAL MODE OF OPERATION.

(1) HIGH SPEED and NORMAL indicator lights are illuminated (Operating and Operating Rate Groups).

(2) Depress the MASTER CLEAR button (Operating Group).

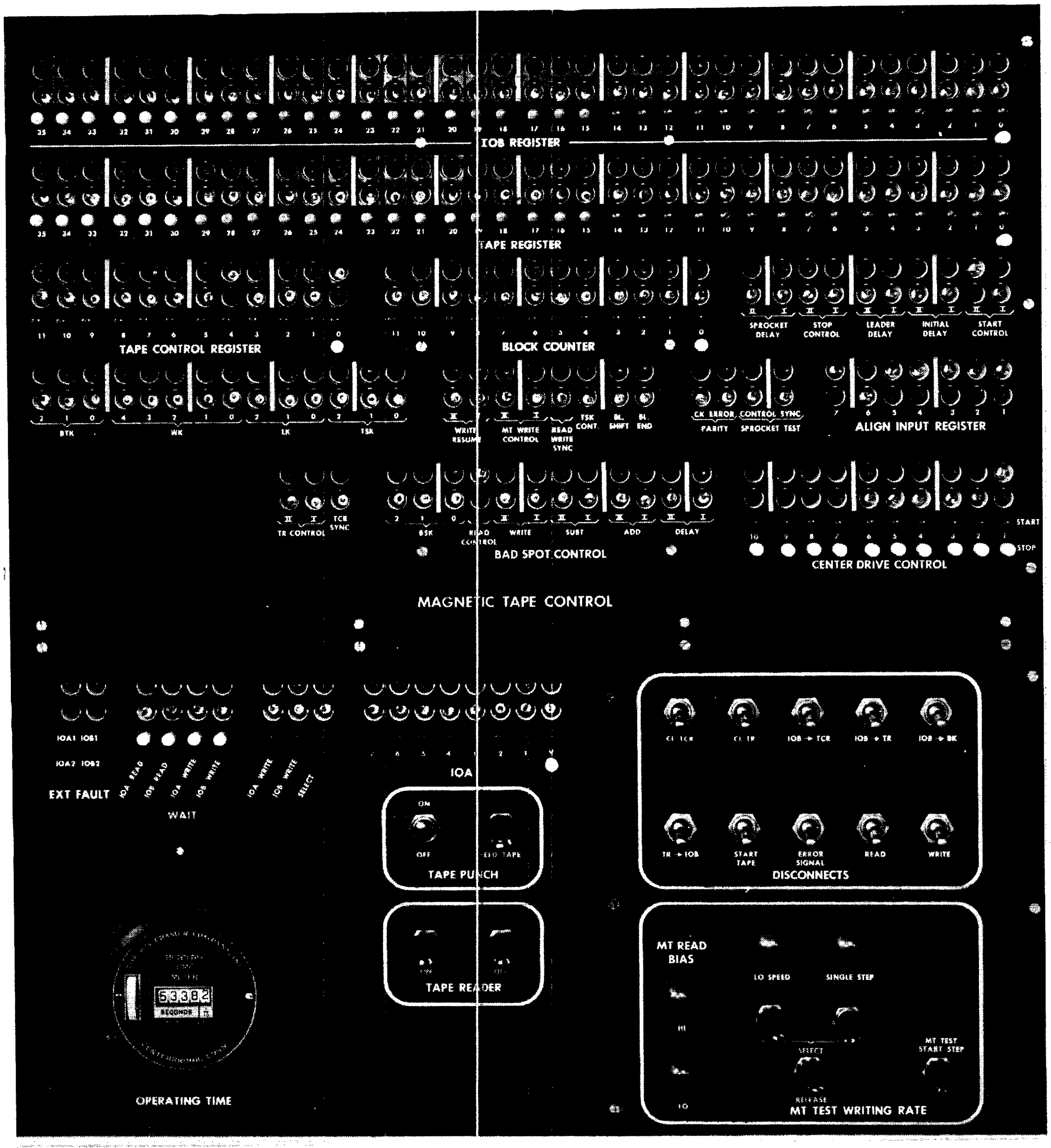
(3) Depress any SELECT JUMP or STOP buttons called for by the program. The associated indicators will illuminate.

(4) Depress the START button (Operating Group). The OPERATING indicator light is illuminated and the lights in Step (a) remain illuminated.



0 10417

Figure 3. Supervisory Control Panel, Right Section



0 10418

Figure 4. Supervisory Control Panel, Left Section

PX 39

If the OPERATING indicator does not illuminate in step (4), check the ABNORMAL CONDITION indicator in the Operating Group. If this is illuminated, one of the disconnect switches is set to its ABNORMAL position. Setting this switch to its NORMAL position will extinguish the ABNORMAL CONDITION indicator and allow a NORMAL mode start.

The depression of the MASTER CLEAR button in step (2) sets PAK to 40000 and MPD to 6. Unless other selections are made by manually depressing buttons on the control panel, the first instruction to be executed will be taken from drum address 40000. Any other manual selections may be made also where buttons are provided after depressing the MASTER CLEAR button and before depressing the START button. After the START button has been depressed, no selections can be made until a computer stop.

If the NORMAL indicator is not illuminated in step (1), the TEST indicator will be illuminated indicating one of the following conditions:

(1) If the HIGH SPEED indicator is not illuminated, one of the MANUAL or AUTOMATIC STEP indicators is illuminated. Depressing the RELEASE button in the Operating Rate Group illuminates the NORMAL indicator.

(2) The TEST/NORMAL switch or the MD NORMAL/ABNORMAL switch (Test Switch Group) is in its "up" position. Setting these switches to their "down" positions illuminates the NORMAL indicator.

All these conditions allow a computer start, but in the TEST mode.

Operation in the NORMAL mode is halted by an A Fault, B Fault, Force Stop, or a programmed stop. Indication of the operation halt is given by the drop of the OPERATING indicator light. If the fault is caused by the selection of an abnormal condition during operation, the ABNORMAL CONDITION indicator will be illuminated.

c. TEST MODE OF OPERATION.

(1) Depress the MASTER CLEAR button (Operating Group).

(2) Select the TEST mode by setting the TEST/NORMAL switch to TEST or the MD NORMAL/ABNORMAL switch to ABNORMAL (Test Switch Group). The TEST indicator (Operating Group) is illuminated, the NORMAL indicator is extinguished, and the HIGH SPEED indicator (Operating Group) remains illuminated.

(3) If other than high speed operation is desired, depress one of the MANUAL or AUTOMATIC STEP buttons (Operating Rate Group). The appropriate indicator is illuminated. (Depression of any of these buttons yields an automatic selection of the TEST mode regardless of whether the TEST/NORMAL switch is set to TEST.)

(4) Depress any SELECT JUMP or STOP buttons called for by the program.

(5) Depress the START button (Operating Group). The OPERATING indicator is illuminated.

(6) Depress the STEP button (Operating Group) if necessary.

If any of the disconnect switches in the Test Switch Group or the MT Disconnect Switch Group are set to their ABNORMAL position, the ABNORMAL indicators (Operating Group and Test Switch Group) are illuminated.

The depression of the MASTER CLEAR button sets PAK to 40000 and MPD to 6. Unless other selections are made by manually depressing buttons on the control panel, the first instruction to be executed will be taken from drum address 40000, or, if the MD NORMAL/ABNORMAL switch is set to ABNORMAL, from the reserve space on the drum. Any other manual selections can be made where buttons are provided after depressing the MASTER CLEAR button. Manual selections can be made from the control panel while in the TEST mode when the computer is in actual operation. One exception to this is that PAK cannot be changed manually when the OPERATING indicator is illuminated.

To change a selection in the Operating Rate Group, computer operation must be at a halt (the OPERATING indicator is extinguished); then, depressing the RELEASE button cancels the previous selection made and allows a new choice of an operating rate.

Operation is halted by an A Fault, B Fault, Force Stop, or a programmed stop and is indicated by the drop of the OPERATING indicator light.

d. JUMP AND STOP SELECTIONS. - The manual selections necessary to effect a programmed Manually Selective Jump, instruction 45 jv, with j = 1,2, or 3, are made by depressing SELECT JUMP buttons in the Selective Jumps Group. To be effective the selections must be made while the computer is not in actual operation (when the OPERATING indicator is extinguished), either NORMAL or TEST mode. An effective manual selection is indicated by the illumination of the SELECTIVE JUMP, 1, 2, and/or 3 indicator. To nullify a selection, the appropriate RELEASE JUMP button is depressed. This also must be done when the computer is not in actual operation.

The manual selections necessary to effect a programmed Manually Selective Stop, instruction 56 jv, with j = 1, 2, or 3, are made by depressing SELECT STOP buttons in the Selective Stops Group. A stop selection may be made during actual computer operation, NORMAL or TEST mode, and is indicated by the illumination of the light immediately above the button depressed. When a stop occurs, it is indicated by the illumination of a SELECTIVE STOP light. (A Manually Selective Stop instruction with j = 0 requires no manual selection.) To cancel a stop selection, the appropriate RELEASE STOP button is depressed. The release of a stop selection may also be made during actual computer operation, NORMAL or TEST mode.

A FORCE STOP selection made by depressing the button so entitled in the Operating Group halts computer operation in either the TEST or NORMAL mode. A force stop is indicated by the illumination of the FORCE STOP indicator light.

e. MANUAL INTERRUPT SELECTION. - An interrupt may be initiated manually by depressing buttons in the Program Interrupt Control Group. Two selections are necessary to activate a line to the interrupt control of the computer. Depressing the ENABLE button allows the line to be energized and illuminates the INDICATE ENABLE LIGHT. Depressing the INITIATE button when the enable is indicated by the light momentarily energizes the line. When the line is energized a signal is sent to the Program Interrupt Control to effect the interrupt during the execution of the next instruction which ordinarily would be concluded by the normal termination commands. Each time an interrupt is desired, the INITIATE button must be depressed and the INDICATE ENABLE indicator must be illuminated. Each time the line to the Program Interrupt Control is energized, the INDICATE ENABLE light is extinguished. Also, depressing the RELEASE button extinguishes the INDICATE ENABLE light and renders it impossible to energize the interrupt line by depressing the INITIATE button.

3. RESTORATION OF OPERATION AFTER STOPS.

The computer ceases operation at the occurrence of a programmed STOP, a FORCE STOP, an EMERGENCY OFF, or a fault condition. The stops by classes are discussed subsequently, and the steps necessary to resume operation noted.

a. PROGRAMMED STOPS

(1) MANUALLY SELECTIVE STOP. - The Manually Selective Stop instruction (56jv) stops the computer operation if the programmed j (0,1,2, or 3) agrees with the selection made on the Supervisory Control Panel (no button selection is provided for j = 0; the computer will always stop in this case). Whether or not a stop occurs at the execution of this instruction, the next instruction will be taken from the v address. When a stop occurs, the OPERATING indicator is extinguished and the appropriate SELECTIVE STOP indicator (red) is illuminated. To resume operation, depress the START button.

(2) FINAL STOP. - The Program Stop instruction (57--) indicates the end of the program. The SELECT STOP indicators and SELECTIVE JUMP indicators, if illuminated, will remain illuminated, and the FINAL STOP indicator (Selective Stops Group) is illuminated. To resume operation, it is necessary to depress the MASTER CLEAR button and follow the procedure for initiating computation in one of the two computer modes.

b. FORCE STOP. - An unscheduled stop of the computer can be effected by depressing the FORCE STOP button (Operating Group). The OPERATING indicator is extinguished and the FORCE STOP indicator (Operating Group) is illuminated. After the condition which prompted the stop has been corrected, operation is resumed by pressing the START button. During operation if it is desired to change any selection in the Operating Rate Group or Selective Jumps Groups, depressing the FORCE STOP button allows the RELEASE buttons in either of these groups to be depressed and a new selection to be made.

c. EMERGENCY STOPS.

(1) MANUAL EMERGENCY STOP. - In cases of extreme emergency, such as a fire, pressing the EMERGENCY OFF button on the Supervisory Control Panel removes <u>all</u> voltages from the equipment. Since all power is removed, a program in process is halted and cannot be immediately resumed. Since such a stop could destroy information, it may be well to reload the program and data into the system after proper operation is restored by maintenance procedures.

(2) AUTOMATIC EMERGENCY STOPS. - This stop also calls for maintenance procedures to restore proper operation.

d. FAULT CONDITIONS.

(1) A FAULT. - An A Fault results in a computer stop which extinguishes the OPERATING indicator and illuminates an A Fault indicator. The specific fault is indicated by the illumination of one of the following indicators, the first six of which are in the A Fault Group on the panel:

DIVIDE

SCC (Storage Class Control)

PRINT

TEMP. (Low Temperature - 100°F)

WATER

OVERFLOW (only possible on Multiply Add instruction, 72 uv)

ABNORMAL CONDITION - Indicated by lights in Operating Group and Test Switch Group

The A Fault does not alter the program in process so that after appropriate corrective action, the operation can be resumed.

Generally, a DIVIDE,SCC, PRINT, or OVERFLOW fault indication is derived from a program error. First, check the appropriate instructions or operands. If these are correct, an actual machine malfunction is the cause and should be isolated and corrected. Operation may be resumed after the Divide, Print, and Overflow faults by depressing the CLEAR A FAULT button and then the START button. If the program requires that the cause of these faults be corrected before continuing the program, operation can be resumed (after the fault correction) with the instruction which caused the fault by inserting the address of this instruction in PAK, setting MPD to 6, and depressing the START button.

An SCC fault results from a reference to some address not permissible under the particular circumstances. The A Fault condition is cleared by depressing the CLEAR A FAULT button, but operation cannot be resumed without correcting the condition that caused the fault. In most cases the faulty instruction in storage should be corrected. After this has been accomplished, the address of the faulty instruction can be inserted in PAK, MPD set to 6, and operation resumed by depressing the START button.

The TEMP. indication is the result of a high air temperature at some point. The high air temperature is indicated by the illumination of one of the amber indicators mounted above the cabinet doors. Corrective measures should be applied immediately unless the urgency for problem results dictates that the BYPASS TEMPERATURE INTERLOCK key switch should be turned. This allows the problem to be continued despite the over-temperature condition. The WATER indication necessitates correcting a water pressure fault (over-pressure or under-pressure condition) before the program can be resumed.

If any of the disconnect switches is accidently or intentionally set to the "up" position after operation has been initiated (in Normal mode only), the ABNORMAL CONDITION indicators in the Test Switch Group and the Operating Group are illuminated, and an A Fault condition occurs and stops computation. Movement of the switch to the normal position will correct the condition.

Operation is resumed after the correction of the Temperature, Water, and Abnormal Condition faults by depressing the START button.

(2) B FAULT. - The B Fault results in a stop which manifests iself much like a Program Stop in that it is necessary to MASTER CLEAR and restart the program. The B FAULT indicator, and one of the indicators listed below, are illuminated in the B Fault Group.

MCT (Main Control Translator)

VOLTAGE

IO (Input-Output)

MT (Magnetic Tape)

MATRIX DRIVE (An MC Fault)

The IO fault indicates incorrect input/output procedure or faulty operation of external equipment. The illumination of the IO Fault indicator and one of the EXTERNAL FAULT indicators located on the left section of the control panel indicates the improper use of IOA or IOB. The EXTERNAL FAULT indicators are as follows:

IOA 1 Read
IOB 1 Read
IOA 2 Read
IOB 2 Read

These faults occur when there has been no indication that the particular register involved is ready for the information being received by it. If the fault is due to an error in program timing considerations, an IOB 1 Read fault indicates that an External Read to transmit input information in IOB to X has not been executed before additional input information is received by IOB from external equipment; and an IOB 2 Read fault indicates that the external equipment has not received the information placed in IOB by an External Write or External Function instruction before IOB receives input information from external equipment. The IOA 1 Read and IOA 2 Read faults indicate similar conditions involving the IOA register.

The IO fault, in conjunction with one of the External Faults above, generally indicates a program timing error. An IO fault resulting from one of the External Faults is cleared by depressing the MASTER CLEAR button. This extinguishes all the associated fault indicators and enables a restart of the program. An IO fault without one of the IOA/IOB faults could also result from a program error. A check of the fault indicator lights on the external equipment being used should reveal the nature of the fault. Faults originating in external equipment because of an incorrect program are cleared by depressing the MASTER CLEAR button. This extinguishes both the IO indicator and the B FAULT indicator and allows a program restart. Faults arising from the improper operation of external equipment are cleared by correcting the condition instigating the fault and then depressing the MASTER CLEAR button. The Master Clear extinguishes the B FAULT indicator. If the IO fault indicator is still illuminated, the source of the fault has not been removed and restarting the program will yield the same fault stop.

Generally an MCT fault is due to a program error (an illegal operation code). In this case the program in storage or the input tape should be checked for accuracy. If no errors are thus discovered, a machine malfunction is indicated and must be corrected. If the fault is due to a program error, the MCT as well as the B FAULT indicators are cleared by a MASTER CLEAR.

The VOLTAGE, or MATRIX DRIVE indication necessitates corrective maintenance, and an MT indication may require corrective maintenance, depending upon the type of MT fault. The MT faults are discussed in the Input and Output section of this volume and are explained in full in the volume supplied with the Magnetic Tape System. The VOLTAGE, MATRIX DRIVE, and some of the MT faults require correction at the source. After these are cleared the specific fault indicator will be extinguished, and the B FAULT indicator will be extinguished by a MASTER CLEAR, allowing a program restart.

To summarize, in general, when a fault is caused by an improper program, its fault indicator as well as the B FAULT indicator is extinguished by depressing the MASTER CLEAR button. If a fault is caused by a machine malfunction, its fault indicator is extinguished by the correction of the fault, and the MASTER CLEAR extinguishes the B FAULT indicator.

Since there may have been errors introduced to the program in process at the time of the fault, it is necessary that a MASTER CLEAR be selected and computation be started anew. If the content of any addresses have been altered by the instructions during execution of that part of a program which was completed before the fault, it would be well to reload the computer before attempting computation.

4. MANUAL READING AND WRITING.

The following procedures in TEST mode are frequently used to manually check portions of a stored program or enter a program into storage. Also included is a procedure for manually transferring a program from Magnetic Drum Storage to Magnetic Core storage.

a. MANUAL WRITING FROM THE Q REGISTER. - The procedure below can be used to alter an existing program in storage or to insert a new program into storage without using a tape reader. The new words are entered into storage via the Q Register by the following steps.

Step 1. Depress MASTER CLEAR (Operating Group)
Step 2. Select MANUAL STEP OPERATION (Operating Rate Group.

Step 3. Set MPD to 0
Step 4. Set MCR to 75 (Repeat instruction)
Step 5. Set UAK to 50000 (set j to 5)
Step 6. Set VAK to 00000
Step 7. Depress START button (Operating Group)
Step 8. Depress STEP button (Operating Group)

Steps 4 through 8 set up an unterminated Repeat Sequence. Since j is 5, only the v address will be advanced at the end of each storage reference.

Step 9. Clear PCR (Clear MCR, UAK, and VAK)
Step 10. Set MCR to 11 (Transmit Positive instruction)
Step 11. Set UAK to Q address
Step 12. Set VAK to first address to be written into
Step 13. Set up in Q Register word to be written
Step 14. Press STEP button
Step 15. Clear Q Register

Steps 10 through 14 manually enter the word set up in Q at the selected v address. To continue writing in consecutive addresses, repeat steps 13, 14, and 15 for each word to be written. If only a single word is to be written, steps 4 through 9 and step 15 can be omitted since a Repeat operation is not needed.

b. MANUAL READING TO THE Q REGISTER. - The procedure below transmits the contents of chosen storage registers to the Q Register with the words at consecutive addresses being displayed on the Supervisory Control Panel.

Step 1. Depress MASTER CLEAR (Operating Group)
Step 2. Select MANUAL STEP OPERATION (Operating Rate Group)
Step 3. Set MPD to 0
Step 4. Set MCR to 75 (Repeat instruction)
Step 5. Set UAK to 60000 (set j to 6)
Step 6. Set VAK to 00000
Step 7. Depress START button (Operating Group)
Step 8. Depress STEP button (Operating Group)

Steps 4 through 8 set up an unterminated Repeat Sequence. Since j is 6, only the u address will be advanced at the end of each storage reference.

Step 9. Clear PCR (Clear MCR, UAK, and VAK)
Step 10. Set MCR to 11 (Transmit Positive instruction)
Step 11. Set UAK to address of first word to be read
Step 12. Set VAK to Q address
Step 13. Depress STEP button

Steps 10 through 13 manually read the word at the selected u address to the Q register where it is displayed for observation. Each time the STEP button is pressed, a word from a consecutive u address will be displayed in Q. If only a single word is to be read, steps 4 through 9 can be omitted since a Repeat operation is not needed.

c. PROGRAM CORRECTION. - If, in reading to the Q Register, an incorrect word is noted, the following procedure is used to insert the correct word at the proper address. (This procedure can be used whether or not a Repeat sequence is being used in the manual reading.)

Step 1.	Depress FORCE STOP button (Operating Group)
Step 2.	Clear Q
Step 3.	Clear UAK and VAK
Step 4.	Set UAK to Q address
Step 5.	Set VAK to address to be written into
Step 6.	Set up in Q Register word to be written
Step 7.	Depress START button (Operating Group)
Step 8.	Depress STEP button (Operating Group)

Steps 1 through 8 enter the correct word into storage. To return to the reading process:

Step 9.	Clear UAK and VAK
Step 10.	Set UAK to next address to be read from
Step 11.	Set VAK to Q address
Step 12.	Depress STEP button

Steps 9 through 12 return control to the manual reading procedure.

d. MANUAL BLOCK TRANSFER. - To effect a manual block transfer from Magnetic Drum Storage to Magnetic Core Storage, the following steps should be performed.

Step 1.	Depress MASTER CLEAR (Operating Group)
Step 2.	Select MANUAL STEP OPERATION (Operating Rate Group)
Step 3.	Set MPD to 0
Step 4.	Set MCR to 75 (Repeat instruction)
Step 5.	Set UAK to 3n (set j to 3 and n to the number of words to be transferred)
Step 6.	Set VAK to a w address containing a 56jv instruction
Step 7.	Depress START button (Operating Group)
Step 8.	Depress STEP button (Operating Group)

Steps 4 through 8 set up a terminated Repeat Sequence. Since j is 3, both the u address and the v address will be advanced at the end of each storage reference.

Step 9.	Depress FORCE STOP button
Step 10.	Release MANUAL STEP OPERATION
Step 11.	Clear PCR (Clear MCR, UAK, and VAK)
Step 12.	Set MCR to 11 (Transmit Positive instruction)
Step 13.	Set UAK to initial MD address
Step 14.	Set VAK to initial MC address
Step 15.	Depress START button

CODING FOR THE COMPUTER

1. SUMMARY OF MACHINE CHARACTERISTICS.

GENERAL		Parallel mode of operation Internal binary number system Two address logic
WORD LENGTH		36 bits (binary digits)
NUMBER NOTATION		"1's complement" binary system
PARALLEL ACCESS REGISTERS (ARITHMETIC SECTION)	A	72-bit Accumulator (A_{71}, A_{70},...,A_0)
	A_R	rightmost 36 bits of A (least significant)
	A_L	leftmost 36 bits of A (most significant)
	Q	36-bit shifting register (Q_{35}, Q_{34},...,Q_0)
	X	36-bit exchange register (X_{35}, X_{34},...,X_0)
PARALLEL ACCESS STORAGE (INDIVIDUALLY ADDRESSED)	RAS	4,096 words of Rapid Access Storage 12,288 words optionally available
	MD	16,384 words of Magnetic Drum Storage
ARRANGEMENT OF MAGNETIC DRUM STORAGE, MD		4,096 bits on each track 4,096 words on a group of 36 tracks 4 groups of tracks, giving 16,384 words Variable interlace between the Storage Address Register and angular location counter permits choice of the angular interval between memory locations having consecutive addresses.

ALLOCATION OF ADDRESSES	RAS	00000-07777 (octal)	4,096 words
		10000-17777 (octal)	4,096 words optional
		20000-27777 (octal)	4,096 words optional
		30000-30777 (octal)	illegal addresses
	Q	31000-31777 (octal)	1 word
	A	32000-37777 (octal)	1 double length word
	MD	40000-77777 (octal)	16,384 words

FIXED ADDRESS ALLOCATION	F1	00000 (octal) in RAS, Rapid Access Storage or 40001 (octal) in MD Storage
	F2	00001 (octal) in RAS
	F3	00002 (octal) in RAS

COMPOSITION OF AN INSTRUCTION WORD

Instruction word	36 bits (i_{35}, i_{34}, ..., i_0)
Operation code	6 bits (i_{35}, i_{34}, ..., i_{30})
First execution address, u	15 bits (i_{29}, i_{28}, ..., i_{15})
Second execution address, v	15 bits (i_{14}, i_{13}, ..., i_0)

SECTIONS OF ADDRESSES

j one-digit octal number represented by u_{14}, u_{13}, u_{12}.
n four-digit octal number represented by u_{11}, u_{10}, ..., u_0.
k number of shifts, represented by v_6, v_5, ..., v_0 or u_6, u_5, ..., u_0.

NOTATION FOR CONTENTS OF REGISTERS

Brackets are used to denote "contents of". Thus:
(u)=36-bit word at address u.
(Q)=36-bit word in Q.
(A)=72-bit word in A.
(A_R)=36-bit word in A_R.
(A_L)=36-bit word in A_L.

DOUBLE-LENGTH EXTENSIONS

D(u)=72-bit word whose right-hand 36 bits are (u) and whose left-hand 36 bits are all alike and equal to the left-most bit of (u).
S(u)=72-bit word whose right-hand 36 bits are (u) and whose left-hand 36 bits are all zero.
D(Q), D(X), S(Q), and S(X) are similarly defined
L(Q)(u)=72-bit word whose left-hand 36 bits are zeros and each of whose right-hand 36 bits is given by the bit-by-bit product of the corresponding bits of (u) and (Q).
L(Q')(v)=72-bit word whose left-hand 36 bits are zeros and each of whose right-hand 36 bits is given by the bit-by-bit product of the corresponding bits of (v) and the complement of (Q).

CONTROL REGISTERS

PAK	Program Address Counter
SAR	Storage Address Register
PCR	Program Control Register
MCR	Main Control Register
UAK	U Address Counter
VAK	V Address Counter

PROGRAM SEQUENCE CONTROL

The complete operation for the execution of a computer instruction consists of two parts.
Part one - the execution of the current instruction, CI.
Part two - the acquisition of the next instruction, NI.

At the start of part one, the Program Control Registers already contain CI as the result of the second part of the previous operation, and (PAK) is y plus 1, where y is the address from which CI was acquired.

During part two, NI is acquired from the address held in PAK at the end of part one, and (PAK) is then increased by one.

Thus, provided that CI does not call for a change in (PAK), NI will be acquired from address y plus 1. In a normal program sequence, successive instructions are obtained from consecutive addresses.

A departure from the normal sequence is called a "jump", and is achieved by altering (PAK) during part one of an operation. Instructions that call for a change in (PAK) are called "jump" instructions.

INPUT-OUTPUT REGISTERS	IOA	An in-out register of 8 stages.
	IOB	An in-out register of 36 stages.
	TWR	A typewriter register of 6 stages.
	HPR	A high-speed punch register of 7 stages.

INPUT DEVICES

Photoelectric Paper Tape Reader
Punched Card Input-Output System
Other optional peripheral equipment

OUTPUT DEVICES

Electric Typewriter
High-Speed Paper Tape Punch
Punched Card Input/Output System
Univac Line Printer
Other optional peripheral equipment

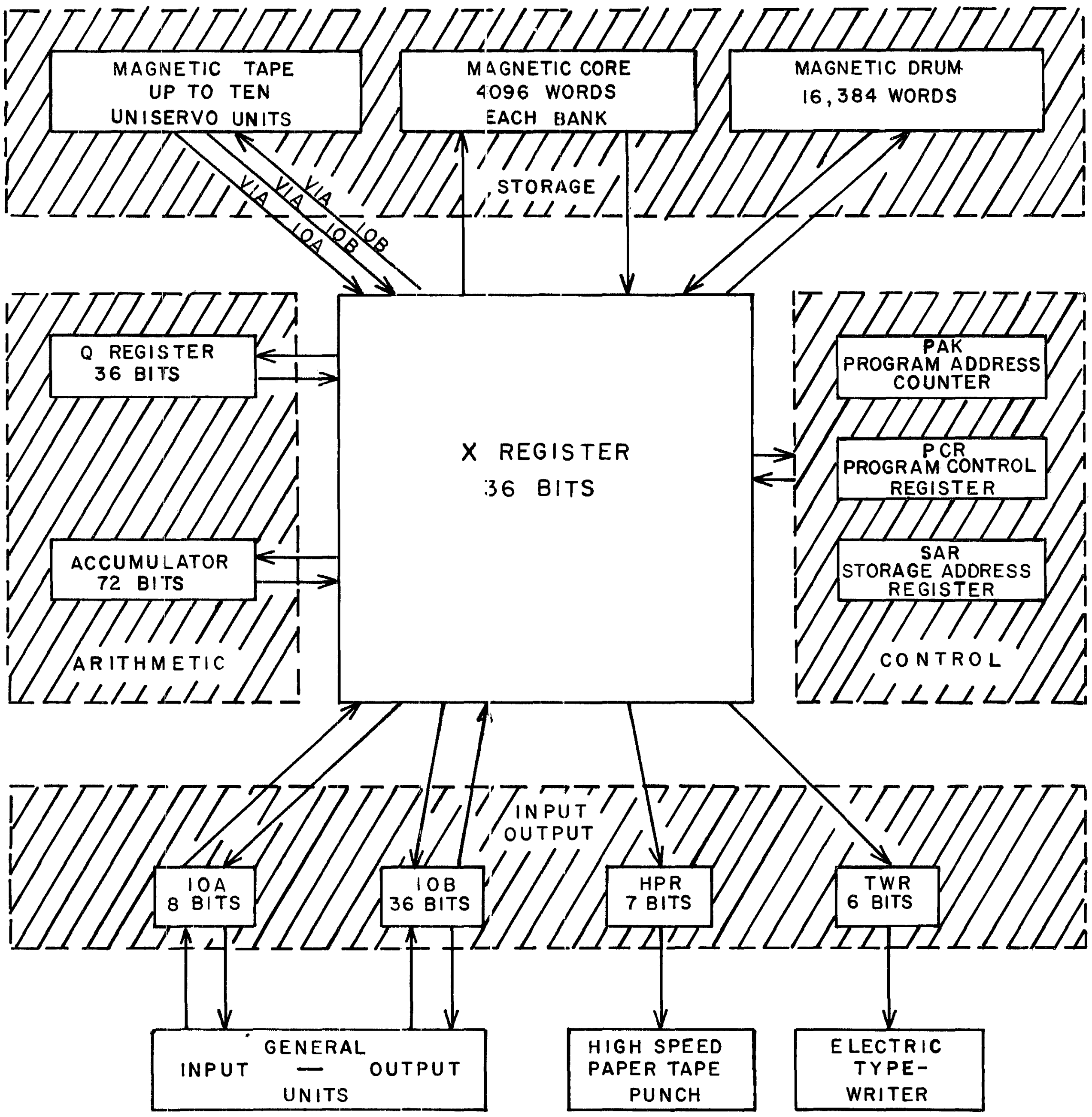

MAGNETIC TAPE UP TO TEN UNISERVO UNITS
MAGNETIC CORE 4096 WORDS EACH BANK
MAGNETIC DRUM 16,384 WORDS
VIA IOA
VIA IOB
VIA IOB
STORAGE
Q REGISTER 36 BITS
ACCUMULATOR 72 BITS
ARITHMETIC
X REGISTER 36 BITS
PAK PROGRAM ADDRESS COUNTER
PCR PROGRAM CONTROL REGISTER
SAR STORAGE ADDRESS REGISTER
CONTROL
INPUT OUTPUT
IOA 8 BITS
IOB 36 BITS
HPR 7 BITS
TWR 6 BITS
GENERAL INPUT — OUTPUT UNITS
HIGH SPEED PAPER TAPE PUNCH
ELECTRIC TYPE-WRITER

Figure 1. Programmer's Simplified Block Diagram

TABLE 1. REPERTOIRE OF INSTRUCTIONS

TPuv	11	TRANSMIT POSITIVE	$(u) \longrightarrow v$
TMuv	12	TRANSMIT MAGNITUDE	$\|(u)\| \longrightarrow v$
TNuv	13	TRANSMIT NEGATIVE	$(u)' \longrightarrow v$
IPxx	14	INTERPRET	$Y + 1 \longrightarrow F_1$, take (F_2) as NI
TUuv	15	TRANSMIT U ADDRESS	$(u_{29-15}) \rightarrow v_{29-15}$
TVuv	16	TRANSMIT V ADDRESS	$(u_{14-0}) \rightarrow v_{14-0}$
EF-v	17	EXTERNAL FUNCTION	Select Ext. Equipment and perform (v)
RAuv	21	REPLACE ADD	$\left[(u) + (v)\right] \longrightarrow u$
LTjkv	22	LEFT TRANSMIT	Shift (A) by k;j=0,$(A_L)_f \rightarrow v$;j=1,$(A_R)_f \rightarrow v$
RSuv	23	REPLACE SUBTRACT	$\left[(u) - (v)\right] \longrightarrow u$
CCuv	27	CONTROLLED COMPLEMENT	$\left[(u) \oplus (v)\right] \longrightarrow u$
SPuk	31	SPLIT POSITIVE ENTRY	$S(u) \longrightarrow A$, Shift (A) by k
SAuk	32	SPLIT ADD	(A) + S(u), Shift (A) by k
SNuk	33	SPLIT NEGATIVE ENTRY	$\left[S(u)\right]' \longrightarrow A$, Shift (A) by k
SSuk	34	SPLIT SUBTRACT	(A) - S(u), Shift (A) by k
ATuv	35	ADD AND TRANSMIT	$\left[(A) + D(u)\right] \longrightarrow v$
STuv	36	SUBTRACT AND TRANSMIT	$\left[(A) - D(v)\right] \longrightarrow v$
RJuv	37	RETURN JUMP	$y + 1 \longrightarrow u$, take (v) as NI
IJuv	41	INDEX JUMP	$\left[D(u)-1\right] \longrightarrow A$; $(A)_f$ +, $(A)_f \longrightarrow u$, take (v)
TJuv	42	THRESHOLD JUMP	$(u) > (A)$, take (v) as NI
EJuv	43	EQUALITY JUMP	$(u) = (A)$, take (v) as NI
QJuv	44	Q JUMP	(Q)+, take (v); (Q)-,take (u); (Q) left 1
MJjv	45	MANUALLY SELECTIVE JUMP	j=0, or j=1,2,3, & MJS=, take (v)
SJuv	46	SIGN JUMP	(A)-, take (u); (A)+, take (v)
ZJuv	47	ZERO JUMP	$(A) \neq 0$, take (u); (A)=0, take (v)

TABLE 1. REPERTOIRE OF INSTRUCTIONS (CONCL'D)

QTuv	51	Q-CONTROLLED TRANSMIT . . .	$L(Q)(u) \longrightarrow v$
QAuv	52	Q-CONTROLLED ADD	$\left[(A) + L(Q)(u)\right] \longrightarrow v$
QSuv	53	Q-CONTROLLED SUBSTITUTE . .	$\left[(L(Q)(u) + L(Q)'(v)\right] \longrightarrow v$
LAuk	54	LEFT SHIFT IN A	$D(u) \longrightarrow A$, Shift (A) by k, $(A)_f \longrightarrow u$
LQuk	55	LEFT SHIFT IN Q	$(u) \longrightarrow Q$, Shift (Q) by k, $(Q)_f \longrightarrow u$
MSjv	56	MANUALLY SELECTIVE STOP . .	j=0, stop; j=1,2,3, & MSS=, stop
FS--	57	FINAL STOP	Stop and indicate
PR-v	61	PRINT	Typewriter performs code in v_{5-0}
PUjv	63	PUNCH	Punch (v_{5-0}); j=1, 7th level also
MPuv	71	MULTIPLY	$(u)(v) = (A)$
MAuv	72	MULTIPLY ADD	$(A)_i + (u)(v) = (A)_f$
DVuv	73	DIVIDE	$(A)_i = (u) \quad (Q)+(A)_f$, $(Q) \rightarrow v$; $(A)_f = +R$
SFuv	74	SCALE FACTOR	Shift D(u) in A until $A_{34} \neq A_{35}$, $(SK) \longrightarrow v$
RPjnw	75	REPEAT	Execute N1 "n" times, jump to F_1
ERjv	76	EXTERNAL READ	j=0, $(IOA) \longrightarrow v$; j=1, $(IOB) \longrightarrow v$
EWjv	77	EXTERNAL WRITE	j=0, $(v) \longrightarrow IOA$; j=1, $(v) \longrightarrow IOB$

2. WRITING A PROGRAM.

a. INTRODUCTION. - The word "coding" is usually used to indicate the preparation of the explicit list of instructions that a computer must execute in order to solve a particular problem. The person who does the coding may be referred to as a "coder". The resultant list of instructions is referred to as either a "routine" or, if the coded product is part of a larger one, a "subroutine". Sometimes the term "programming" is used to refer to the process described above as coding. Ordinarily, however, "programming" refers to a more inclusive process, which includes not only the coding but the final stages of formulation of the problem for the computer; i.e., the numerical analysis, the selection of computational procedures, the specification of input-output formats, etc.

b. INSTRUCTION NOTATION. - In the following text the mnemonic notation is used to indicate the operation code of an instruction. For example, "TP" is used instead of "11" to indicate the Transmit Positive instruction; "MP" is used instead of "71", etc. The address portions of the instruction are indicated by octal numbers. For example, the composite symbol

```
TP     01012     01013
```

denotes the Transmit Positive instruction calling for the transmission of the contents of address 01012 to address 01013. If this were written in octal, it would appear as

```
11 01012 01013.
```

In binary, it would appear as

```
001 001 000 001 000 001 010 000 001 000 001 011.
```

As illustrated above, it would be impractical to code using the binary system. For this reason, and because the conversion of numbers from base 2 to base 8, and vice versa, is immediate, the octal number system is used in coding.

When preparing a routine, not only the instructions which are to be executed must be specified, but also the placement of these instructions in the computer must be indicated. The storage location of an instruction in the computer is referred to as the storage address of the instruction. It is customary to indicate this address to the left of the notation for the instruction, as follows:

```
01010     TP     01012     01013.
```

It is common practice when displaying routines for exposition to give some explanatory comment to the right of the instruction. In the following routine the left column contains the location in storage. The central column contains the contents of that location; for instance, an instruction or datum. The right column indicates the author's reason for including this word in the routine.

Routine for replacing the data at 00101, 00102, - - -, 00110
by the first backward difference of this data.

```
00010     RS     00110     00107     form backward difference

00011     RS     00010     00015     decrease (00010)

00012     EJ     00014     00000     has (00010) reached (00014)?

00013     MJ     00000     00010

00014     RS     00100     00077     terminal dummy instruction

00015     00     00001     00001     constant (address decrement)
```

c. LOOPS. - The above is an example of a routine which has a "loop". The loop in this case consists of the three instructions at 00010, 00011 and 00012. The first of these forms one of the backward differences and stores it where the minuend of the subtraction was. The second instruction, the Replace Subtract at address 00011, then modifies the instruction which forms the backward difference by decreasing both of its addresses by 1. Thus, the next time the loop is traversed and a backward difference is formed,it will be the difference of the contents of the decreased addresses. The next instruction, the Equality Jump, tests to see whether or not the loop has been traversed a sufficient number of times. If equality exists between the most recently modified content of address 00010 and the dummy instruction at address 00014, the task has been completed and an exit occurs to address 00000. If equality does not exist, the next instruction at address 00013 is executed. This instruction provides a jump back to the beginning of the loop at 00010, and the whole process is repeated once more.

The above example illustrates one very important facet of coding; a number of the instructions in any routine are always involved only indirectly with the desired results. In this example, only one instruction, the first, produces directly the first backward difference. The others are concerned with keeping the process going, advancing addresses, and testing to see if the process has been completed. Such instructions are frequently referred to as "housekeeping" instructions. The functions of address modification, terminal testing, and jump provisions are referred to as "housekeeping" functions.

It is common practice to indicate the fact that certain instructions are altered during the course of the computation by putting brackets around the instruction or that part of it which is altered. Thus, the first line of the above example could have properly been written as

```
00010     [RS     00110     00107]
```

or as

```
00010     RS     [00110]     [00107]
```

The latter is less preferable then the former because it suggests that the addresses are independently modified, which is not true in this case. Bracketing the so-called modified instructions, or modified addresses, serves to call attention to the fact that at some later time during the computation, the addresses may not be as they appear on the page. In particular, in the example above, at the end of the subroutine the addresses at 00010 will have been modified to agree with those appearing at storage address 00014.

Addresses which are modified within a loop should generally be set to their desired initial values by instructions prefacing each loop, i.e., the process of "prestoration"; or by instructions following each loop, i.e, the process of "restoration". (Restoration is less preferable than prestoration because the restoration instructions may not be executed if a computer fault is incurred during the loop.) If this is done the routine can be used any number of times while it is in the memory without reloading it from a permanent storage device. Note that the above example does not meet these specifications. Below is the same example coded with such "self-restoring" properties. In this routine the loop consists of the instructions at 00010, 00011, and 00012. Note that a different means is provided to terminate the loop. Note also the use of boxing to indicate a loop, and the use of a straight line, following an unconditional jump, to indicate a break in the sequential acquisition of the next instruction to be executed.

Routine for replacing the data at 00101, 00102, . . ., 00110
by the first backward difference of this data.

```
00006     TP     00017     00016        Prestore counter, (00016)

00007     TP     00014     00010        Prestore (00010)

00010    [00     00000     00000]       Form backward difference

00011     RS     00010     00015        Decrease (00010)

00012     IJ     00016     00010        All differences formed?

00013     MJ     00000     00000        Exit
-------------------------------------------------------------------
00014     RS     00110     00107        Initial contents of 00010

00015     00     00001     00001        Constant (address decrement)

00016     00     00000     00000        Counter, initially n-1.

00017     00     00000     00006        Constant, n-1, initial value
                                        of counter.
```

In this case the termination test is performed by an Index Jump instruction which is executed after each traversal of the loop. Execution of the Index Jump at address 00012 causes a "1" to be subtracted from the contents of 00016 and a test to be made to see if the result is negative. If the result is not negative, a jump to address 00010 is made for another traversal of the loop. Memory location 00016 is called a "counter" since it contains at each stop a count-down tally of the number of traversals of the loop. If the Index Jump instruction is at the end of a loop (which is to be traversed n times), this counter should have an initial value of n-1. The quantity is a constant stored at location 00017. Entry of the routine at address 00006 causes this quantity n-1 to be transferred to the counter at location 00016. The termination test could also have been coded at the beginning of the loop. In this case, however, the initial value of the counter should be "n" to effect n traversals of the loop.

When the termination test indicates that the loop has been traversed n times, the execution of the Index Jump instruction at address 00012 does not cause a jump but allows the next instruction to be taken from 00013. The exit instruction, a Manual Jump, is stored here.

Any of the conditional jumps, Index Jump, Threshold Jump, Equality Jump, Q Jump, Sign Jump, or Zero Jump, can be used as the decision instruction to determine the number of times a loop has been traversed.

In summary, a loop consists of four elements.

(1) A series of self-restoring operations which set up the loop for the first traversal. These initial "housekeeping" instructions place in a "counter" a number which determines the number of times, n, that the computer will traverse the loop. The instructions also set in the routine the initial addresses of any data referenced by the instructions within the loop.

(2) The computational instructions which lead to the solution of the particular problem.

(3) The "housekeeping" instructions which advance/decrease the addresses which reference the data so that successive loop computations are performed on successive data.

(4) The "housekeeping" instructions which advance/decrease the counter, and the decision instruction(s) which test the counter to determine if the loop has been traversed n times.

d. SUBROUTINES.

(1) INTRODUCTION. - A portion of a routine which is complete in itself and can be isolated from the context of the larger routine is known as a "subroutine". A subroutine is a self-contained list of instructions for executing some particular operation. If it contains the calculations necessary to compute a function such as $\sqrt{x}$, sine x, tan x, etc., it should be coded in such a way that it may be used in a number of routines wherever such a function is desired.

(2) SUBROUTINES WITH PARAMETERS. - In the case of subroutines which are to be used in many different programs, absolute addresses may not be assigned to certain quantities. These addresses are regarded as parameters of the subroutine and chosen in accordance with the main program. In such a case it is conventional to write, in lieu of an absolute address, some designation for the quantity itself enclosed in braces. Thus, "$\{x\}$" means "the address at which the quantity x is stored". For example, the instruction for

transmitting x to address 01033 would be written

$$\text{TP} \quad \{x\} \quad 01033.$$

The following subroutine for adding two vectors together exemplifies the use of parameters. The two vectors are denoted by x and y with coordinates x_i and y_i, respectively. The sum vector is denoted by z with coordinate z_i. Thus, it is necessary to form the sums

$$z_i = x_i + y_i \qquad i = 1, 2, 3, - - -, n.$$

00100	TP	00114	00117	set counter to n-1
00101	TU	00112	00104	set in $\{x_1\}$
00102	TU	00113	00105	set in $\{y_1\}$
00103	TV	00112	00105	set in $\{z_1\}$
00104	TP	$[\{x_i\}]$	A	form
00105	AT	$[\{y_i\}]$	$[\{z_i\}]$	$z_i = x_i + y_i$
00106	RA	00104	00116	Advance i by 1
00107	RA	00105	00115	
00110	IJ	00117	00104	termination test
00111	MJ	00000	[00000]	exit
00112	00	$\{x_1\}$	$\{z_1\}$	constant
00113	00	$\{y_1\}$	00000	constant
00114	00	00000	n-1	constant
00115	00	00001	00001	constant
00116	00	00001	00000	constant
00117	00	[00000	00000]	counter

This subroutine assumes that tabular values of x_i and y_i are stored in consecutive order somewhere in the memory and that a table of values of z_i is to be stored in consecutive order in the memory. The constants located at 00112 and 00113 are, in effect, parameters of this subroutine because these constants give the addresses of x_1 and y_1, and the address where z_1 is to be stored.

Another parameter of this subroutine is the terminal value of i (i=1, 2, 3, - - -, n). The subroutine has a loop from 00104 to 00110 which needs to be traversed n times.

(3) SUBROUTINE ENTRANCE AND EXIT. - To make use of a subroutine, the main routine must provide a jump to the correct entry point of the subroutine, and the subroutine must provide an exit back to the main routine. In the subroutine for adding two vectors the entry address is 00100. The v address portion of the Manual Jump instruction at 00111 must be set to a jump-out address in the main routine.

The simplest method of entering a subroutine is to use the Return Jump instruction. To enter the above subroutine the instruction RJ 00111 00100 would be written at address y in the main routine. Execution of this instruction causes the quantity y + 1 to be placed in the v-address portion of 00111 and the contents of 00100 to be taken as the next instruction. Thus, a return to the main routine at address y + 1 is provided upon completion of the subroutine. In this way the subroutine may be entered from many different points in the main routine.

If a subroutine has parameters, these must be set up before it is executed. The parameters may be inserted by instructions in the main routine. Thus, in the subroutine for adding two vectors, the contents of addresses 00112 and 00113 could have been entered by instructions in the main routine. If the parameters occur in many different instructions throughout a subroutine, the subroutine itself will contain the instructions for inserting the parameters in all necessary locations. Again in the subroutine for adding two vectors, instructions 00100 through 00103 distribute the parameters throughout the routine. Since each parameter is used only once, it would be as easy for the main routine to set the parameters directly in addresses 00104 and 00105 as to place them in addresses 00112 and 00113. However, when a parameter occurs more than once in a subroutine, it is better that the main program insert the parameter in a single location and the subroutine distribute it from there. This avoids the possibility of a programmer omitting some of the parameter settings.

A convenient means of repeatedly executing a subroutine with a different parameter each time is illustrated next. The subroutine is referenced by a Return Jump instruction followed immediately by the list of parameters. Following the last parameter is the next instruction to be executed after the repeated use of the subroutine.

y	37	00010	00011
y+1	00	a_1	n_1
y+2	00	a_2	n_2
.	.	.	.
.	.	.	.
.	.	.	.
y+k	00	a_k	n_k
y+k+1	NI		

Here the parameters a_i and n_i, i = 1, ..., k, are for the following subroutine for punching n + 1 consecutive words at addresses a_j, j=i, i+1, ..., $i+n_i$.

Subroutine for Punching n + 1 Consecutive Words

```
00010   MJ   00000   [00000]   Exit
00011   SP   00010    00017    Extrance
00012   TU     A      00013    y+1 --> address of 00013, (y+1) = P_1
00013   TP   [P_i]      Q      P_i --> Q
00014   QT   00036      A    \ If (A)=0, this is a parameter; go to 00021
00015   ZJ   00016    00021  / If (A)≠0, this is NI; go to 00016
00016   SP   00013    00071  \
                              > j+k+1 --> v address of 00010
00017   TV     A      00010  /
00020   MJ   00000    00010    to exit instruction
00021   TU     Q      00023    a_i --> u address of 00023
00022   TV     Q      00037    n_i --> 00037
00023   TP   [a_j]      Q      (a_j) --> Q; j = 1, i+1, - - -, i+n_i
00024   TP   00040    00041    set counter to punch 5 frames
00025   LQ     Q      00006    shift to punch next two octal digits
00026   PU   00000      Q      punch
00027   IJ   00041    00025    have 5 frames been punched?
00030   LQ     Q      00006    shift to punch last two octal digits
00031   PU     Q        Q      punch with 7th level to show end of word
00032   RA   00023    00042    increase (00023) to pick up next word
00033   IJ   00037    00023    have n_i+1 words been punched?
00034   RA   00013    00042    increase (00013) to pick up next parameter
00035   MJ   00000    00013    return to pick up next parameter
00036   77   00000    00000    mask to detect parameter
00037  [00   00000     n_i]    counter for number of words to be punched
00040   00   00000    00004    initial value of counter, 00041
00041  [00   00000    00000]   counter for number of frames per word
00042   00   00001    00000    constant to augment u address
```

The Return Jump instruction at address y places y + 1 in the v-address portion of 00010. Therefore the subroutine can recover from 00010 the address of the first parameter, P_1. Instructions 00014 and 00015 test the content of y + 1 to see if it is a parameter or if it is the next instruction. This test is based on the fact that there is no operation code "00". When "NI" is detected, the exit in 00010 is to y+k+1.

Note that this device permits listing any number k of parameters. The mechanics may easily be varied to accomodate sets of parameters of any form, e.g., sets of parameters which occupy more than one word. Different methods of detecting the end of the parameter list are possible.

The Univac Scientific instruction repertoire includes the Interpret instruction for referencing subroutines with parameters. An Interpret instruction at address y places y + 1 in the v address of F_1 and takes (F_2) as the next instruction. In other words, the operation code of the Interpret instruction in itself indicates the same operation as the instruction RJ 00000 00001. The advantage of the Interpret instruction lies in the availability of 10 octal digits in which information may be stored. Address F_2 contains a jump to a subroutine which uses the information in the Interpret instruction as parameters for its operation. The subroutine can set up its own exit from the v address of F_1.

Frequently, complex systems of pseudo-coding are built around the Interpret instruction. These "interpretive systems" simplify the job of programming for a computer because of the use of pseudo-instructions whose functions define operations which are actually carried out by a series of ordinary machine instructions. Such series of stored instructions are in the form of stored subroutines, the operations of which are initiated by the interpretation of the pseudo-instructions.

Chief among the Interpretive systems are those providing for arithmetic operations on floating binary point numbers. (See subparagraph c.) In a typical system, the Interpret instruction is composed as follows, each character denoting an octal digit:

$$\text{IP} \quad \text{RA} \quad u_3 \quad u_2\ u_1\ u_0 \quad v_3\ v_2\ v_1\ v_0.$$

Here the pseudo-code "RA" refers to the Replace Add subroutine, corresponding to the machine operation of Replace Add, for floating binary point numbers. Each of the operand addresses u and v has four octal digits rather than five. Since only four octal digits are needed to span the range of addresses in RAS, this is not a severe restriction. The interpretation routine deciphers the third and fourth octal digits, in this case RA, to determine which of the subroutines in the system is to be executed.

(4) LIBRARY OF SUBROUTINES. - In any computational laboratory having a Univac Scientific, there is a library of subroutines which can be used to calculate the frequently encountered mathematical or logical operations required by the particular laboratory. These are then available for any programmer to use in coding his own programs. In addition to saving coding time, library routines have the advantage of being "debugged" and thoroughly tested. Presumably, more effort has gone into their design than could be afforded by individual

programmers. On the other hand, library routines may be more general then necessary for a particular program, and hence wasteful of time and storage.

Subroutines may be grouped into three basic categories. The following listing is intended to give the beginning coder an idea of the type of subroutines that would be contained in a typical library of subroutines. This list is by no means intended to be complete.

(a) COMPUTATIONAL SUBROUTINES.

Basic arithmetic:

- Floating point add, multiply, divide, etc.
- Complex number operations
- Multiple precision routines

Function evaluation:

- Trigonometric, sin x, cos x, tan x, etc.
- Exponential, $f(x) = x^a$, a^x, etc.
- Square root
- Hyperbolic functions

Numerical Analysis:

- Solution of n simultaneous equations
- Matrix operations
- Numerical integration
- Numerical differentiation

Logical:

- Sorting
- Collating
- Boolean algebra

(b) SERVICE SUBROUTINES AND/OR ROUTINES.

- Compilers
- Assembly
- Programming aids (debugging subroutines)
- Machine testing routines

(c) INPUT-OUTPUT SUBROUTINES.

- Punch card (read and punch)
- Paper type (read and type)
- Editing (typewriter, paper tape, or card output)

The need to become thoroughly familiar with the available library of subroutines cannot be over-emphasized. An understanding of what each subroutine will accomplish and the requirements for its use is important.

(5) ASSEMBLY OF SUBROUTINES. - If each of the library subroutines were coded for a specific location in the computer memory, a programmer wishing to use subroutines would be faced with two unpleasant alternatives: (1) code his main routine around the fixed locations of the subroutines, or (2) manually recode the subroutines to operate from locations which he chooses.

Instead, the practice is to adopt some standard form for subroutines and to use an "assembly program" to convert the standard form of the subroutine to coding suitable for execution at a specified location.

The simplest of these assembly routines assumes that the standard form of the subroutine is already at the desired location. It remains for the assembly routine to modify addresses (within the instructions of the subroutine) relative to that location.

A more sophisticated assembly routine provides not only for address modification but for transfer of the subroutine to a designated location from a library file on magnetic tape or magnetic drum.

Furthermore, there are in use "compiling" routines which relieve the programmer of assigning storage locations to subroutines. The programmer need only refer to a subroutine by means of an identifying index. The compiler routine then arranges the subroutines in storage and modifies all subroutine references to provide for a jump to the assigned location.

One standard form for subroutines in use at several computer installations is as follows.

(a) The initial address of each subroutine is address 01000 in rapid access storage. The service routines (assembly, compiling, etc.) treat address 01000 as a "relative" address.

(b) The first instruction of the subroutine provides for the "alarm" exit. Most subroutines are coded to detect the unexpected or undesirable, such as entry with an out-of-range argument or incorrect parameters. When such is detected, a jump to 01000 occurs. At 01000 is the instruction

```
37     76000     76002
```

which provides entrance to an "Alarm Print Routine". This routine prints (A), (Q), and the address of the instruction at which the alarm condition occurred.

(c) The second instruction provides for the normal exit from the subroutine. The address of this instruction, 01001, is the u address of the RJuv instruction used to enter the subroutine. This instruction is a Manually Selective jump, coded:

```
01001     MJ     0(=j)     [30000].
```

Actually the v address of this instruction of 30000 is an unallowable address which will cause an SCC computer fault if the jump to 30000 occurs. Normally the RJ instruction used to enter the subroutine would replace the v address of the MJ instruction with y+1. However, if the RJ instruction had been incorrectly coded, or the subroutine had been entered without an RJ instruction or without first changing the v address of (01001), the computer would be stopped after executing the subroutine when (01001) is taken as the next instruction. Thus would an incorrect entrance into the subroutine be indicated.

(d) The third instruction of the subroutine (01002), is the entry line into the subroutine. This address, 01002, is the v address of the RJ instruction used to enter the subroutine. If more than one entry into a subroutine is necessary, instructions referencing the various desired entry points in the subroutine are usually stored at consecutive memory positions beginning with the third line of the subroutine.

(e) A jump instruction must be placed at the end of the subroutine computations to cause a jump to 01001, the normal exit line of the subroutine.

(f) All instructions and "modifiable" operands should be placed in consecutive memory positions. "Modifiable" operands are any words referencing storage locations of the subroutine which would be changed upon translation of the subroutine. The "unmodifiable" operands should be stored directly following the above, at consecutive locations.

(g) The subroutine must contain all the housekeeping instructions that make it self-restoring.

The Square Root Routine given below is a routine coded in the standard form just described. This routine is to be entered with the argument in the Accumulator. The square root of that argument is in the Accumulator upon exiting from the routine. Note the jump at 01002 to the alarm exit should the argument be negative, and the jump at 01007 to the alarm exit should the argument be out-of-range.

<u>Routine for Evaluating the Square Root of N</u>

Entrance conditions:	$(A_L) = 0$ $(A_R) = N = \overline{N}\cdot 2^{34}$
Exit conditions:	$(A_R) = \sqrt{N} = \sqrt{\overline{N}}\cdot 2^{17}$

The method used here is the Newton-Raphson iteration

$$X_{n+1} = X_n + 1/2\left[\frac{N}{X_n} - X_n\right] \quad (= X_n + \Delta X_n)$$
$$= 1/2\left[X_n + \frac{N}{X_n}\right]$$

The first approximation is $X_0 = 2^{35-1}$ and convergence is assumed when $\Delta X \geq 0$.

01000	37	76000	76002	Alarm Exit
01001	45	00000	[30000]	Normal Exit
01002	46	01000	01003	Entrance; $\overline{N} \cdot 2^{34}$ negative?$\longrightarrow$Alarm
01003	11	01016	00005	$2^{35}-1 \longrightarrow (t_5)$
01004	11	A	00003	$\overline{N} \cdot 2^{34} \longrightarrow (t_3)$
01005	43	01016	01001	$\overline{N} \cdot 2^{34} = 2^{35} - 1 \rightarrow$ Exit
01006	36	00003	A	$\overline{N} \cdot 2^{34} - (A_R) \longrightarrow (A)$
01007	47	01000	01010	$(A) \neq 0 \longrightarrow$Alarm
01010	31	00003	00041	$1/2\ \overline{N} \cdot 2^{68} \longrightarrow (A)$
01011	73	00005	00004	$1/2\ \overline{N}/x_i \cdot 2^{34} \longrightarrow (Q), (t_4)$
01012	54	00005	00107	$1/2\ x_i \cdot 2^{34} \longrightarrow (A), (t_5)$
01013	23	Q	00005	$(1/2\ \overline{N}/x_i - 1/2\ x_i) \cdot 2^{34} \longrightarrow (Q) = \Delta x$
01014	21	00005	00004	$1/2\ x_i \cdot 2^{34} + 1/2\ \overline{N}/x_i \cdot 2^{34} = (x_i + 1)\ 2^{34} \longrightarrow (t_5)$
01015	44	01010	01001	Δx negative, Repeat loop
01016	37	77777	77777	$2^{35}-1$

Note the use of addresses 00003, 00004, 00005 (t_3, t_4, t_5) as temporary storage for partial results obtained during computation.

Most routines require working storages such as these during their operation. Upon completion of the routine the contents of these locations are no longer of use. A single location may be used several times in the same routine for temporary storage of different quantities. For instance, in this routine 00005 contains successively x_i, $1/2\ x_i \cdot 2^{34}$ and $x_{i+1} \cdot 2^{34}$. Since it is possible for many different subroutines to use the same locations for temporary storage, it is conventional to reserve an area of the memory for temporary storage. Several installations have reserved RAS locations 00000-00037 for this use.

In this routine the words in which addresses are to be modified when the subroutine is assembled are in 01000 to 01015. The word in 01016 is a constant and is not to be modified.

An example follows of an assembly routine which is suitable for modifying subroutines in the previously given standard form. Recall that each subroutine to be assembled must be coded in absolute form as if its initial instruction were at address 01000. In actual fact the subroutine is placed in

RAS at any location selected by the coder. It is the function of the assembly routine to modify all addresses that are dependent on the location of the subroutine. The assembly routine must be informed of the actual RAS addresses of the initial instruction of each subroutine and of the number n of successive instructions to be modified in each case. It will then scan (s+j-1) instructions, j=1, 2, ... n, altering the u and v addresses of the instructions by adding s-01000 to them when necessary.

A particular address will be modified if, and only if, the 10th bit from the right in its 15 bit array is a one; i.e., if it is of the form XXX XX1XXX XXX XXX. For RAS addresses this means that all addresses of the form 01xxx (x an arbitrary octal digit) and only such addresses will be modified. Thus, addresses of temporary storage such as those described previously (addresses 00000 through 00037) will not be altered. Since n successive instructions are modified starting with the initial one, constants should be stored at addresses 01000 + n and following.

The routine below will assembly k subroutines in succession. It is assumed that the parameters for each, n_i and s_i, are of the form

$$00 \quad n_4\, n_3\, n_2\, n_1\, n_0 \quad s_4\, s_3\, s_2\, s_1\, s_0$$

and are stored consecutively at addresses 00121 + i, i = 1, 2, ... k. Address 00121+k+1 must contain 0 to indicate the end of the parameter list.

<u>Address</u> <u>Modification</u> <u>Routine</u>

00100	11	[00122]	A	(A) = (00121 + i)
00101	47	00102	00000	Exit -- All modified
00102	73	00073	00005	(00005) = n_i
00103	16	A	00112	$(00112)_v = s_i$
00104	36	00121	00006	(00006) = s_i -01000
00105	41	00005	00110	Exit for next subroutine
00106	21	00100	00073	Advance i by one
00107	45	00000	00100	Jump to 00100
00110	16	00112	00116	$(00116)_v = (00112)_v$
00111	11	00040	00007	(00007) = zero
00112	21	00007	[00000]	(00007) $=(s_j)_i$
00113	55	A	00033	Shift right nine binary digits
00114	51	00075	Q	(Q) = 00 00001 00001

```
00115     71     Q         00006     (A) = increment to (sj)i
00116     35     00007    [00000]    (sj)i = modified initial (sj)i
00117     21     00112     00074     sj advanced by one
00120     45     00000     00105]    Jump to 00105
          ____________________
00121     00     00000     01000     Constant 1000 octal
```

Note the references to addresses 00040, 00073, 00074, 00075. The contents of these addresses are commonly used constants.

```
00040     00     00000     00000
00073     00     00001     00000
00074     00     00000     00001
00075     00     00001     00001
```

A computer installation will sometimes reserve an area of RAS for a "constant pool", i.e., a collection of frequently used constants. All subroutines can then refer to the constant pool rather than having each contain its own constants. This results in a considerable saving of storage space when many subroutines are used. In the particular case of this assembly routine, it is assumed that there is a constant pool at addresses 00040 through 00077.

In the case of routines, such as the assembly routine, which are to be used by many different persons, it is important that the routine be accompanied by complete and easily understood operating instructions. The operating instructions for the previous assembly routine are given below.

Operating Instructions for the Assembly Modification Routine.

1 Only suitably coded subroutines may be assembled. A subroutine is suitably coded if it satisfies the following conditions.

a It is written in absolute form as if the initial instruction were located at address 01000.

b All addresses independent of the location of the routine are of the form XXX XX0 XXX XXX XXX.

c All addresses to be modified are of the form XXX XX1 XXX XXX XXX.

d All subroutine constants follow immediately the last instruction.

2 To use this routine:

a Load all subroutines to be modified in their proper locations in RAS _b_ Load this assembly program in 00100-00121

_c_ Load the constant pool in 00040-00077

_d_ Load the parameters

$$00 \quad n_i \quad s_i \qquad i = 1, 2, \ldots k$$

for all k subroutines into addresses 00121+i, i = 1, ..., k

_e_ Load zero into address 00121 + k + 1

_f_ Provide for the proper exit in the u address of 00101 and enter at address 00102.

The assembly routine which has been described is an example of a service routine designed to relieve a coder of part of the work in preparing a program for execution by the computer. The text following describes a coding technique (simplifying the job of coding) which is possible because of another type of service routine.

e. RELATIVE ADDRESSING. - In coding routines it is often preferable not to assign absolute memory locations to quantities referred to in the program. For instance, it may be convenient to postpone the assignment of absolute addresses to data, constants, temporary storage and the like, until the coding of the entire problem has been completed. Similarly, it may not be desirable to assign locations to subroutines until coding is completed and it is known what the storage requirements are for each of the sections of the program.

It is possible to postpone the assignment of absolute addresses by coding with "relative" addresses. For instance the instruction

```
01010    11    01012    01013
```

which is coded with absolute addresses 01010, 01012, and 01013 might be written with relative addresses as

```
C10    11    C12    C13.
```

In this case, the addresses are relative to the address C. The alphabetic character C could denote any address allowable in the Univac Scientific. For example, it could be assigned the address 01000. The numerals following an alphabetic character in a relative addressing scheme usually are interpreted as being additive i.e., if C denotes 01000, then C12 denotes 01012, etc. Almost any of the alphabetic characters are usually allowable in most relative addressing schemes. A few exceptions are the reservations of the letter A, denoting the Accumulator, and the letter Q, denoting the Q register. Other exceptions to the use of alphabetic characters need not concern the beginning coder. The allowable characters depend upon the conventions in use at a particular computer establishment.

Each new alphabetic character used in a program may indicate a new "region" in the storage of the computer. In fact, a greater number of regions may be accomodated by using combinations of alphabetic, or alphabetic and numeric characters. By using relative addresses it becomes comparatively simple to assign segments of a problem to various regions with the routines assigned to each region performing a separate calculation or function.

As an example of a routine coded with relative addresses, consider the following.

Compute the function f(y), where $f(y) = \frac{5y-2}{y+7}$.

Assume that $y+7 \neq 0$ and $|y+7| < 2^{35}$.

Region		Instruction		Function
(Program of instructions)	Operation code	u address	v address	
C1	TP	f1	A	Place y in A
C2	AT	E1	f2	Place y + 7 in f2
C3	TN	E3	A	Place -2 in A
C4	MA	E2	f1	Form 5y-2 in A
C5	DV	f2	f3	Place the result of 5y-2/y+7 in f3
C6	PS	0	0	Stop
(Constant Storage)				
E1	0	0	7	
E2	0	0	5	Constants
E3	0	0	2	
(Temporary Storage)				
f1	-	-	-	Variable y
f2	0	0	0	Temporary storage holding y+7
f3	0	0	0	Temporary storage holding f(y)

After a program is coded with relative addresses, it must be converted to absolute addresses before execution by the computer. The computer may be used to perform this conversion. "Translation" routines have been written which not only convert relative addresses to absolute addresses but which also convert decimal information to computer binary representations. Frequently these translation routines include an assembly subroutine.

f. MECHANICS OF CODING.

(1) FLOW DIAGRAMS. - After it has been established that a particular problem can be solved by the computer, it is necessary to formulate the problem in terms of the language of the Univac Scientific computer. A program of instructions, and the necessary data needed for the solution of the problem must be devised.

A "flow diagram" is helpful in facilitating the coding or programming of the problem. A flow diagram or flow chart indicates the "flow" or steps in the computation which lead to the solutions of the particular problem.

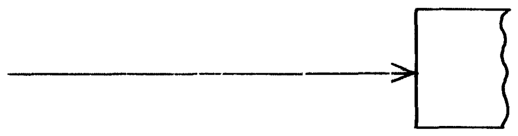
A basic flow diagram usually lists the series of simple arithmetic steps which are to be performed by the computer. It is imperative that the coder be thoroughly familiar with the overall operations and peculiarities of each computer instruction so that he can construct the outline with regard to the capabilities of the computer. A description of each instruction is presented in the section of this volume entitled Sequential Presentation of Instructions. Tabular information on the instructions and the contents of the arithmetic registers before and after the execution of each instruction is presented in the volume entitled "Content of Registers".

Usually more than one flow diagram is formed for the more complicated problems. The first flow outline may be nothing more than equations in mathematical language, written in the sequence in which they will be computed, together with brief explanations of the steps involved. The second flow chart usually formulates the flow of computation as the problem will be computed on the computer. This chart will usually contain the instructions necessary for the data input, the instructions that operate on the input data to obtain the solutions, and the necessary instructions for the output of the results.

Many times the problems are of such a nature that the second type of flow diagram will consist of many charts and/or diagrams, each a more detailed presentation of the preceding charts.

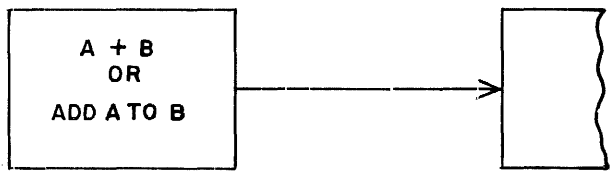
To facilitate the task of forming flow diagrams and to allow other programmers to understand a particular flow diagram, certain symbols have been more or less standardized. The following list of symbols is by no means complete, but is intended to give the coder an example of the basic symbols used in drawing the second type of flow chart.

(a) LINES OF FLOW.



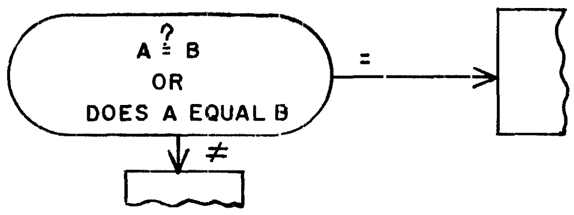
A solid line with an arrow touching the next element of the flow diagram is usually used to indicate the path to be followed by the computer; or more precisely, the path to be followed by the coder who is formulating the computer instructions from the flow diagram.

(b) OPERATION SYMBOL.

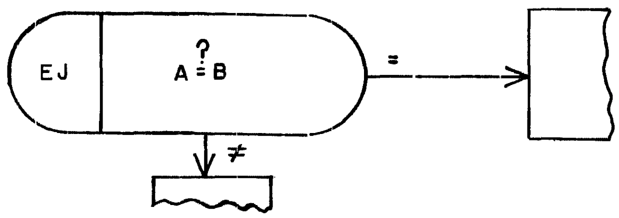


The rectangular box usually contains a statement about a computer or mathematical operation. The contents of the box may be a simple statement or a mathematical expression.

(c) DECISION SYMBOL.

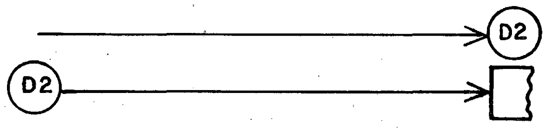


The symbol above is used to indicate a two-way decision. This symbol is sometimes written as



where the letters EJ designate the use of the Equality Jump instruction to make the decision in the computer.

(d) CONNECTORS AND REMOTE CONNECTORS.



To eliminate as much as possible the crossing of lines of flow on a diagram, the above symbols are used to indicate a destination not easily reached in the diagram. Thus, the flow can be broken at a convenient point by terminating it in an arbitrary symbol which can be used to initiate the flow in another region of the paper.

(e) EXAMPLE. - To illustrate the use of the above symbols, consider the following problem.

Compute the function $f(x_i)$ where

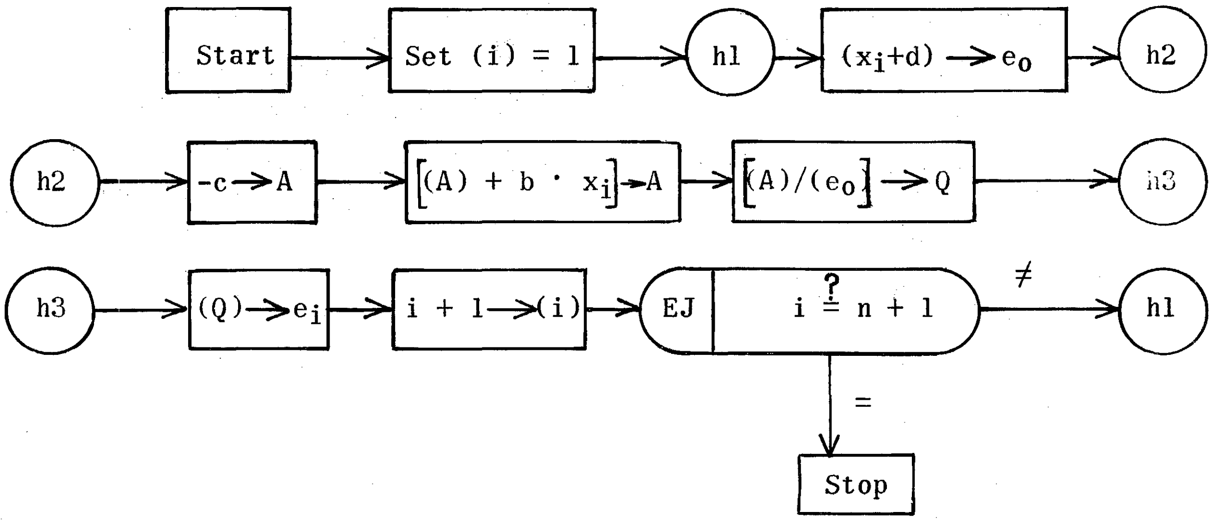
$$f(x_i) = \sum_{i}^{n} \frac{bx_i - c}{x_i + d} \qquad i = 1, 2, 3, \ldots, n \ .$$

For this problem, it is assumed that b, c, d, and x_i are integers and are contained in the computer in the Rapid Access Storage. Also, it is assumed that

$$x_i + d \neq 0$$
$$\left| x_i + d \right| < 2^{35}$$

The function $f(x_i)$ is stored at the location whose address is ei. The symbol (i) indicates the storage of the "i" term.

FLOW DIAGRAM



(2) CODING STEPS. - In summary, there are usually four phases in the preparation of a coded computer program:

(a) The construction of a "flow diagram" or outline which shows the general computational steps to be accomplished;

(b) The creation of a program, written in terms of computer instructions, using numeric or alphabetic symbols to indicate operation codes and the u and v address portions of the instructions;

(c) If the program created in Step 2 is coded in octal, the program is ready for execution after being put onto some input medium and loaded into the computer. If the program is coded in symbolic notation, it must be translated into a binary representation before it can be automatically executed by the computer. Service programs for this translation may be used to facilitate this process by allowing the computer to perform the conversion;

(d) The "debugging" (computer check-out) of the final program by means of trial runs on the computer.

The last phase - debugging - is described in the following paragraphs.

g. DEBUGGING A PROGRAM.

(1) INTRODUCTION. - After the coded program has been written out completely and the manuscript carefully reviewed, the program must be put on some input medium for loading into the computer. The input mediums and the devices which transfer information from these mediums to the computer memory are described in the section Input and Output of this volume.

Once the program has been stored in the computer in binary form, either directly from the input medium or by means of some translation routine, it is ready for execution by the computer. However, if the program is lengthy, it is likely that the coded program as stored has errors. Three common types of errors are:

(a) Tape or card preparation errors made in the preparation of the program for input;

(b) Coding errors, such as listing incorrect or incomplete addresses, transposition of digits in the addresses or operation codes, transcription errors, etc;

(c) Logical errors; incomplete or erroneous methods used to obtain the solution(s).

"Debugging" is the term applied to the process of locating errors in a program and correcting them. Debugging a program usually involves a series of trail runs of the program on the computer. Each time the program fails to run properly, the failure must be analyzed and the error corrected. Frequently one can immediately discover the error, correct it manually from the control console, and proceed with the next trial run. At other times the detection of errors is more difficult and requires a thoughtful reconsideration of the program, keeping in mind any clues as to the origin of the error which may have been provided during the run. Error detection may be facilitated by the use of service routines coded expressly for this purpose.

The methods that can be employed to debug a program will vary widely from installation to installation depending on the nature of the programs, the service routines available, the availability of computer time, and the personal preference of the individual for one procedure over another. The following remarks suggest possible patterns to follow in debugging.

(2) TAPE OR CARD PREPARATION ERRORS. - This type of error can be minimized by systematic checking of all input tapes and cards before their information is read into the computer. The usual method that is used to prepare input paper tapes is listed below. (A similar routine is used in punched card preparation.)

1 A paper tape is punched from the coder's manuscript using the Flexowriter.

2 The resulting typed manuscript from the typewriter is discarded.

3 A new typed manuscript is prepared from the punched paper tape.

4 This typed manuscript is compared with the coder's manuscript to detect errors in punching.

5 All detected punching errors are corrected and a new corrected tape is prepared.

6 From the corrected punched tape a new manuscript is obtained which should be retained by the coder to use while debugging.

7 If seven-level codings are used they should be sight-checked before an attempt is made to read the tape into the computer.

(3) MANUAL DEBUGGING. - Manual debugging is the process which is used to locate errors in a program by controlling the operations of the program from the Supervisory Control Panel and visually checking these operations as they occur and are indicated on the control panel. Although manual debugging does not make efficient use of computer time, it can, if done correctly, reduce the overall time required to debug a program because of the versatility afforded by a "thinking" programmer at the control panel.

After loading the program in the computer, a run at high speed is usually tried. (For a discussion of the various speeds at which the computer may run, see the section Operating the Computer. Many of the suggestions for program debugging require a knowledge of operating the computer from the control console. This is also described in Operating the Computer.) This run may show the program to be free of errors, in which case there is no debugging to be done; or this run at high speed may end immediately with a computer fault. Frequently an MCT (Main Control Translator) fault or SCC (Storage Class Control) fault will arise. The former indicates an illegal operation code; the latter an illegal address. (Computer faults are described in Operating the Computer.)

When a fault occurs, the contents of PAK (Program Address Counter) should first be noted. This will almost always indicate the exact location of the erroneous instruction.

The SCC fault usually arises from either a transcription error, a punching error, or incorrect modification of an instruction by the program. These faults are ordinarily easy to correct. An MCT fault may be caused by a punching or transcription error. Frequently it will result from an erroneous program jump. In this case the address in PAK may indicate that the jump was to an area which is not occupied by the program. This fault must be traced to the instruction which caused the jump.

The computer may also stop on a DIVIDE fault or an OVERFLOW fault. Erroneous coding may cause these faults, or they may be the result of an error in judgement concerning the range of the numbers involved in the computation.

Obviously, the first phase of debugging is to correct the program so that it will run without computer faults.

Although the program is running without incurring computer faults, errors in its execution may still be detected. For instance, the program may be coded to produce output but no printing or punching occurs. Correction of this type of error may be facilitated by observation of the monitoring oscilloscope located on the upper center section of the Supervisory Control Panel. This oscilloscope displays a point for each of the 4096 registers in RAS every time one of the registers is referenced. If, while debugging a program, the scope displays a distinct non-terminating repeated pattern (one dot or many) for a questionable period of time, it can be assumed that the computer is executing an erroneous "loop". This usually means that one or more of the "decision" instructions has an incorrect jump reference causing a return to itself or to a series of instructions which lead back to the incorrectly-addressed jump instruction.

It may be possible to detect from the oscilloscope the area in which such a loop is operating; or it may be desirable to FORCE STOP the computer, select MANUAL STEP OPERATION, and, by executing one instruction at a time, trace the operation through the loop. By noting each jump to a new address and traversing the loop at least once, the incorrect jump will probably be located.

Thus, the second phase of debugging is to eliminate any non-terminating loops.

After these two phases of debugging a program have been completed, output from the corrected program will reveal if the program is being executed properly. If the output format is in error, an investigation of that portion of the program devoted to output is necessary. This may be accomplished by entering the output routine with a typical result and, if the output routine is short, executing the routine one instruction at a time, observing the results at each step. If the output routine is complicated, it will probably be necessary to inspect the manuscript of the coded output routine, using the erroneous results as a guide in looking for the trouble. (Output routines from the subroutine library would not be expected to cause any difficulty.)

In summary, the third phase of debugging is to correct any output routines used in the program.

Since frequent output during the running of a program is a positive indication of proper or improper running, it is often convenient to provide for output of partial results of computation. Since these results are of no interest when the program is known to be running properly, and since any output from the computer is time consuming, routines providing such output should be entered via an optional jump. During the debugging process the optional jump switches may be "on" to provide the output for monitoring purposes. After the program is completely checked, the optional jump switches may be "off", thus eliminating the intermediate output. An intermediate printout may even be of value when the program is in actual use. If normal output is very infrequent, or onto a medium which is not easily read, an operator may wish to occasionally sample the intermediate output as a check on the program's operation.

Since the problem in debugging a program is to isolate each of the errors, it is desirable to have the coding arranged in small segments. The coder should be sufficiently familiar with his program to anticipate the results of each of the segments of the computation. In particular, he should be able to recognize an incorrect result.

A convenient way to examine the results of these program segments is to terminate each with an OPTIONAL STOP. Like the optional jumps described above, these stops may be "on" while the program is being debugged and then "off" when the program has been checked out. If optional stops have been coded in the program, the program may be executed from one stop to another until trouble is encountered. Then, because it is known in which segment the error lies, that section may be re-run on MANUAL STEP OPERATION.

The points in a program at which these optional stops and jumps are placed are called "breakpoints".

Thus, the fourth phase of debugging makes use of programmed breakpoints to isolate errors.

The instructions for optional stop and optional jump each provide three selections which incur the jump possibility. Although the number of options may be increased by combinations of the selections, an option at all desirable breakpoints can not be provided in a lengthy program. Regardless of how judiciously the breakpoints have been selected, in an actual debugging process, a breakpoint is often desired where an option for one has not been coded in the program. In this case it is possible to superimpose a breakpoint on the program. If a stop, for instance, is desired at address a and addresses b and b+1 are unused, the instruction at a may be replaced with the instruction

```
56    00000    b.
```

The program instruction which was in a is then stored in b, and the instruction

```
45      00000     a+1
```

is stored in b+1.

Thus, the fifth phase of debugging is to insert breakpoints if programmed breakpoints are inadequate.

Thus far the process of debugging has been described in terms of the programmer himself manually debugging his routine.

It cannot be overemphasized that in order to do manual debugging, the coder must be thoroughly familiar with the problem, its coding, and in addition, with the peculiarities of the computer.

(4) DEBUGGING WITH SERVICE ROUTINES. - A number of service routines exist which facilitate the process of program debugging. Consider phase 1 of the process of debugging, the elimination of computer faults. Two types of errors may be hard to find: (1) an SCC fault arising from improper modification of an instruction and (2) an MCT fault arising from an incorrect jump. The search for either of these errors may be facilitated by an "address sort" routine. Such a routine scans all instructions of the program for a given address. Any instruction containing that address is printed with its location. In case one the trouble may be located by a search for references to the modified instruction; the faulty jump in case two may be located by a search for references to (PAK)-1 at the time of the MCT fault.

A service routine of use in eliminating faulty loops is a "jump trace" routine. A "trace" or "automonitor" is a routine which executes interpretively all the instructions of a program which is being debugged. As each instruction is executed, the instruction itself and its address are printed out together with (A), (Q), (u), and (v). In the case of a jump trace routine, print-out occurs only if the instruction is a jump. With this print-out, the sequence of jumps in a program can be followed, and those which are incorrect can be noted.

When a program has been run, a "memory dump" will often reveal the operations which occurred during the run. A memory dump routine lists octal printouts of the contents of those memory locations containing the program. The examination of indices, temporary storage, modified instructions, etc., usually reveals the portions of the program which were improperly executed. This aids in locating the improper modifications of the program and recovering partial results. A "changed word post-mortem" routine is a selective memory dump in which only those words which have been changed in the course of running a program are printed. A changed word post-mortem routine requires that the original program as loaded be stored in the memory. After the program is run, a word-by-word comparison is made of the original program and the executed program. The advantages of a changed word post-mortem routine over a memory dump routine are that it operates faster, and there is less print-out to be examined. The memory dump, by virtue of printing all the program, gives evidence of misloading or punching errors which may not be detected by a changed word post-mortem.

The use of breakpoints in debugging a program can be facilitated by any one of several "breakpoint routines". The simplest of these merely replaces selected instructions with manual stops and stores separately the instructions and the jumps back into the program. Upon reaching one of these stops in the program, the programmer may examine the contents of certain locations which contain partial results, or manually step through the next part of the program. A more elaborate breakpoint routine may provide for automatic print-out of the contents of certain registers at a breakpoint. Usually these print-outs are either in octal or decimal notation. It is also possible to invoke at a breakpoint a trace routine which prints the results of each of the next few instructions. A great saving in time is effected by using these sampling and tracing routines at a breakpoint rather than manually examining the results at that point.

The debugging service routines which have been described are of two types, "static" and "dynamic". The static type is employed after a program has been run and has stopped. The memory jump and address sort routines are examples of static routines. The dynamic routines operate in conjunction with the program as it is run. These may be "executive" routines which execute interpretively the program instructions and provide printouts of the results, i.e., trace routines; or they may be routines to which control is transferred at specific breakpoints, i.e., sampling routines. The static debugging routines usually require less computer time for their operation than the dynamic type. Therefore, in practice they are used more often although more information can be obtained from the dynamic type.

(5) ERROR CORRECTION. - It may be possible during the debugging process to correct a routine manually by making simple alterations in the contents of certain memory locations. After these corrections have been made a new input paper tape or punched card is usually needed for reloading the corrected version of the routine. To obtain the corrected version, which is in the memory of the computer, a "punch storage for reload" or "bioctal dump" service routine may be used. This produces a paper tape or deck of punched cards containing the program coded in octal with appropriate insert and check addresses.

In order to conserve computer time while making a number of manual corrections from the Supervisory Control Panel, the following method is suggested:

Depress MASTER CLEAR button

Select MANUAL STEP OPERATION

Set MPD to 3

Set MCT to 75 (Repeat instruction)

Set UAK to 70000 (set j to 7)

Set VAK to 00000

Set X to 11 10000 v, where v is the first address to be written into.

Depress START button

Depress STEP button

$$\left.\begin{array}{l}\text{Set in Q Register the word to be written}\\[1ex]\text{Depress STEP button}\\[1ex]\text{Clear Q Register}\end{array}\right\}\text{Repeat Manually}$$

It should be noted that a j of 7 is being used which sets up an unterminated repeat sequence with both the u and v addresses of the repeated instruction being advanced upon each execution. Since this is the case it becomes comparatively simple to read back into Q for checking purposes the words just written. This procedure is

Clear UAK

Set UAK to u where u is the first address which was written into.

Clear VAK

Set VAK to 10000

$$\left.\begin{array}{l}\text{Depress START button}\\[1ex]\text{Depress STEP button}\end{array}\right\}\text{Repeat Manually}$$

If the data that is to be written or to be read is not in consecutive memory locations, the following steps must be added at the end of both of the above repeated steps.

Clear VAK (UAK if reading)

Set VAK (UAK if reading) to the address to be written into or read from.

The process of program debugging has been described in terms of the programmer himself operating the computer. Manual debugging clearly requires a thorough knowledge of the program. However, the service routines may be applied by an operator with perhaps instructions from the programmer. The extent to which programmers will operate the computer for program debugging will vary from one installation to the next.

h. OPERATING PROCEDURE. - The service routines which have been described as aids in debugging should be available for use whenever trouble arises. Since the need for these routines cannot always be anticipated, it is desirable that they be at all times in the computer memory. Many installations have reserved areas on the magnetic drum and/or magnetic tape for permanent storage of the service library. Except when necessary, programs are not stored in these areas. This practice eliminates the need for frequent reloading of the service routines.

In addition to the service routines for debugging there are service routines which facilitate operating the computer. Chief among these, in the case of a computer with a Photoelectric Paper Tape Reader, is the program for reading information from paper tape. Such a program may be coded to accept information punched in any form whatsoever. In particular, the program might be designed to load bioctal tapes with the 7th level control configurations described in the section Input and Output Systems.

<u>Routine for Loading Bioctal Tapes with a Paper Tape Reader</u>

00000	45	00000	00006	Jump to start
00001	45	30000	00003	Optional jump to bypass stop
00002	17	00005	00002	Stop reader
00003	11	00035	40000	Enter data
00004	21	00003	00037	Advance address
00005	45	30000	00007	Optional jump to bypass start
00006	17	00006	00006	Start reader
00007	76	00000	Q	Read to Q
00010	31	00035	00006	Isolate data levels
00011	52	00027	00035	
00012	31	00036	00001	Isolate seventh level
00013	52	00030	Q	
00014	51	00030	00036	
00015	43	00032	00001	Test for enter data
00016	43	00031	00021	Test for insert address
00017	43	00033	00023	Test for check address
00020	45	00000	00007	

00021	16	00035	00003	Insert address
00022	45	00000	00007	
00023	11	00003	A	Check address
00024	36	00034	A	
00025	43	00035	00007	
00026	57	77777	77777	
00027	00	00000	00077	Data levels mask
00030	00	00000	17700	Seventh level mask
00031	00	00000	11100	Insert address code
00032	00	00000	10100	Enter data code
00033	00	00000	10500	Check address code
00034	11	00035	00000	Constant for check address
00035	--	-----	-----	Word assembly
00036	--	-----	-----	Instruction code assembly
00037	00	00000	00001	Constant for advance address

If the program to be loaded is short enough to allow the storage of a copy, it is helpful to have an "image" of the program stored on the magnetic drum (MD). Usually the image on MD contains the two extra instructions

```
RP   jn   w
TP   MD   RAS
```

which cause the program to be transferred to RAS at the start of the computations. Thus, whenever it is necessary to restart the program, the unaltered version of the program may be transferred from MD without reloading the computer from paper tape or cards. This drum image is also available for use by service routines such as a "changed word post-mortem" routine.

A paper tape loading routine may be used which automatically provides such a drum image of the program. Of greater importance, such a loading routine permits loading anywhere in RAS; the previously listed loading routine does not allow RAS addresses 00000 through 00037 to be loaded. Provided that the loading routine contains an "end of tape" detection, the loading routine may be supplemented to operate under the following conditions:

(a) The loading routine is stored on the magnetic drum,

(b) The loading routine proper is prefixed with instructions which,

   (1) store the current contents of RAS as a drum image, and

   (2) "bootstrap" the loading routine proper into RAS for operation.

(c) All data being loaded into RAS under control of the loading routine is in fact loaded in the drum image.

(d) Upon detecting "end of tape", control is transferred to suffixed instructions (on the drum) which transfer the drum image to RAS and stop the computer.

The coding for such a bootstrap operation follows.

```
L0     11   00000   70000   ]  store F1
L1     75   37777   L3      }  store (RAS) in drum image
                                 70000 - 77777
L2     11   00001   70001   ]
L3     75   3  n    00000   ]  transfer loading routine proper
L4     11   L5      00000   ]    to RAS for operation
L5     45   00000   Lx         entrance to loading routine in RAS
L6                          ]
---                         }  loading routine proper
Ln+4                        ]
Ln+5   75   37777   Ln+7    ]  transfer of drum
Ln+6   11   70001   00001   }  image to RAS
Ln+7   11   70000   00000   ]
Ln+10  56   00000   L0         stop; re-entry to loading routine.
```

It is desirable that a bootstrap procedure similar to this be used with most service routines so that operation of the service routine does not destroy the status quo of RAS.

Another practice which many consider helpful is to "erase" the computer memory before loading a program for debugging. This means that erroneous jumps to an address not used by the program result immediately in an MCT fault. Also program sections separated by erased portions of RAS are easily recognized in a memory dump. This erasing, or writing of zeros in every location, may be effected by a repeated Transmit Positive instruction from Q with (Q) = 0. This may be set up manually, as previously explained, or included in the service library.

Frequently service librarys include routines to simplify manually reading information out of storage or writing information into storage. If these routines print out the word read or entered, a written record is provided of all corrections made in the course of running a program.

When a library of subroutines is kept on magnetic tape, the service library usually includes a routine to extract the subroutines from file. Input translation routines may also be in the service library.

It is important that a programmer be thoroughly familiar with any service routines in use at his installation. There are restrictions to the use of many of these routines of which he must be aware. For instance, the "Routine for Loading Bioctal Tapes with a Paper Tape Reader" cited previously does not permit loading in addresses 00000-00037.

Before coding any routine, a programmer should acquaint himself with the operating conventions of his installation. He should know such particulars as the following:

(a) The service routines available and the facilities and restrictions of each;

(b) The input translation routines in existence,

(c) The procedures necessary in order to use library subroutines: their assembly, their reference to a constant pool, a temporary storage pool, an alarm print, etc., and

(d) The areas, if any, of the computer memory which are reserved for special purposes.

3. NUMBER NOTATION.

a. INTRODUCTION. - Because the theory of operation of the computer is based on binary logic, the coder should become thoroughly familiar with one's complement binary arithmetic. The following list of publications is recommended as references to supplement the discussion of binary arithmetic presented in the section General Description and Appendix A of this volume.

(1) Stifler, W. W. Jr., (ed), High Speed Computing Devices by the staff of Engineering Research Associates, pp. 74-99, (McGraw-Hill, New York, 1950).

(2) Richards, R. K., Arithmetic Operations in Digital Computers, pp. 1-22, (D. Van Nostrand Co., Inc., New York, 1955).

(3) Uspensky, J. V., Elementary Number Theory, (McGraw-Hill, 1939).

As stated previously, for convenience octal, instead of binary notation, is used to describe computer words and addresses. It should be remembered, however, that the logic of computer operation is based on the binary system.

b. RADIX CONVERSION. - In most problems the data for a program is presented in some form other than binary. In particular, data is frequently decimal. In Appendix A procedures for converting numbers from one radix (base) to another are described. It is usually desired that the computer perform the necessary radix conversions of input and output data.

If, for example, the computer were used for a problem involving large amounts of decimal data, an efficient method for converting from decimal to binary would be necessary. A "scalar product routine", such as that which follows, could be used for this conversion provided it is known that all decimal data are integers.

Let N be the desired binary number; then

$$N = a_0 + 10a_1 + 10^2\, a_2 + \dots + 10^n a_n.$$

To form this scalar product, the individual products of the binary-coded decimal digits of the decimal number and their corresponding powers of ten, binary-coded, are accumulated.

<u>Routine to Convert Decimal Integers to Binary Integers.</u>

```
e1        RS      A       A      }  Clear A to zero
e2        RP      3,n+1   e4     }
e3        MA      u1      v1     }  Form N
e4        --      --      --     }  Next instruction
u1                        1      }  10^0
u2                        12     }  10'
.                         .      }  .  Powers of ten coded
.                         .      }  .  in binary.
un+1                             }  10^n
v1                        a0     }
v2                        a1     }
.                         .      }  Binary - coded decimal digits
.                         .      }
.                         .      }
vn+1                      an     }
```

The result N (in binary) appears in the Accumulator.

The reverse operation, conversion from binary to decimal, can be accomplished with a repeated division. Assuming that the positive integral number N to be converted to decimal is contained in the Accumulator, the steps are:

```
f1      RP     3,n+1     f3         Form binary - coded decimal digits

f2      DV     u1        v1

f3      --     --        --         Next instruction

u1                              ⎫   10^n
.                        .      ⎪   .             Powers of
.                        .      ⎬   .             ten coded in binary
.                        .      ⎪   .
un                       12     ⎪   10^1
un+1                      1     ⎭   10^0
v1                       a_n    ⎫
v2                       a_n-1  ⎪
.                        .      ⎬
.                        .      ⎪   Binary - coded decimal digits
.                        .      ⎪
vn+1                     a_1    ⎭
```

The Divide instruction (DVuv) causes the number in the 72-bit Accumulator to be divided by the number at address u. The quotient is deposited at address v, and a positive remainder is left in the Accumulator. In the above example, descending powers of ten are brought in as successive divisors of the diminishing remainder, then the series of quotients are the binary-coded digits of the decimal representation of the number initially contained in the Accumulator, These quotients are depositied at a series of consecutive addresses v_1, v_2, ... v_{n+1}.

To illustrate binary to decimal integer conversion, consider the following.

Convert n binary words from any n consecutive memory locations to decimal, and print these decimal integers on the typewriter with appropriate sign. This subroutine assumes that the Q register, upon entry into the subroutine, contains a code word which specifies the number n and the address of the first binary number; i.e.,

(Q) = 00 xxxxx -----

where ----- specifies the number n and xxxxx specifies the address of the first datum.

```
Y0    MJ   0      30000    Exit line of subroutine
Y1    PR   0        b 6    Shift typewriter to lower case
Y2    TU   Q        Y 7    Set in initial location of data
Y3    TV   Q        b 2    Set in counter n
Y4    IJ   b2       Y 6    Test n:  if n=0, no numbers are printed
Y5    MJ   0        Y 0    Exit - completion of routine
Y6    PR   0        b 1    Perform carriage return
Y7    TP   x_i      A      Place i^th number to be printed in A
Y10   SJ   Y11      Y14    Test sign of number
Y11   PR   0        b 3    Print negative sign
Y12   TM   A        A      Put absolute value of number in A
Y13   MJ   0        Y15    Skip next instruction
Y14   PR   0        b 4    Print plus sign
Y15   PR   0        b 5    Print space
Y16   RP   3,13     Y20  } Form binary-coded
Y17   DV   b7       Y22  }  decimal digits
Y20   RP   2,13     Y22    Form print instructions
Y21   RA   Y22      b 0
Y22                   0  }
Y23                   0  }
Y24                   0  }
Y25                   0  }
Y26                   0  } Print 11 decimal digits for each number.
Y27                   0  } (Initially the binary-coded decimal digits
Y30                   0  } are stored here until they are replaced
Y31                   0  } by the appropriate print instructions.)
Y32                   0  }
Y33                   0  }
Y34                   0  }
```

```
Y35   MJ   0      Y4
b0    PR   0      d0       Dummy print command
b1                45       Flexowriter code for carriage return
b2                 0       Counter n, zero initially
b3                56       Flexowriter code for negative sign
b4                54       Flexowriter code for positive sign
b5                04       Flexowriter code for space
b6                57       Flexowriter code for shift to lower case
b7                -        10^10
b10               .        10^9
b11               .        10^8
b12               .        10^7
b13               .        10^6
b14               .        10^5   Binary-coded powers of ten
b15               .        10^4
b16               .        10^2
b17               12       10^1
b20               1        10^0
d0                37
d1                52
d2                74
d3                70
d4                64       Typewriter codes for the decimal
                           digits, 0 through 9.
d5                62
d6                66
d7                72
d10               60
d11               33
```

Attention is called to the four instructions (Y17), (Y20), (Y21) and (Y22). The first two of these instructions form by repeated divisions the binary-coded decimal digits of each number to be printed. The second pair of instructions add to each of these binary-coded decimal digits a dummy print instruction which addresses in its v address portion the starting address of a table of typewriter codes for the digits 0 through 9. For example, suppose that the first binary-coded decimal digit is a five. This is stored at Y22 as the result of the first division. Then, as a result of the Replace Add instruction at Y21, the content of Y22 could be replaced by PR 0 d5. Thus, when this instruction is executed by the computer, the digit "5" would be printed by the typewriter.

c. SCALING. - It should be emphasized that the above routines are for binary/decimal conversion of <u>integers</u>. Words in the Univac Scientific are considered to be integers in the sense that the computer has been designed so that for all arithmetic operations the binary point is thought to be to the right of the right-most bit. This does not mean, however, that only computations with integral numbers can be performed in the computer.

Whenever a problem involves a quantity s which either (a) exceeds the range of integers which may be represented in a 36 bit word $(1-2^{35} \leq s \leq 2^{35}-1)$, or (b) has a fractional part, that quantity s may be represented as $s = s_1 \cdot 2^{s_2}$, where s_1 is an integer within the range of a computer word. The quantity 2^{s_2} is called a "scale factor", s_1 is called the "mantissa" of the number s, and s_2 is called the "characteristic" of s. A positive characteristic is used to represent integers outside the range $(1-2^{35}, 2^{35}-1)$; a negative characteristic is used to express numbers with fractional parts.

Thus, in a computer operating upon integers, the machine representation of a number is the actual value of the number multiplied ("scaled") by some constant 2^i. It must be remembered that when the machine executes an arithmetic operation involving scaled numbers, the scaling may change. Suppose s_m and t_m are the machine representations, each scaled 2^{20}, of s_e and t_e. The result r_m of a Multiply instruction MP $\{s_m\}$ $\{t_m\}$ is then the machine representation of $r_e = s_e \cdot t_e$ scaled 2^{40}:

$$s_m = s_e \cdot 2^{20} \qquad t_m = t_e \cdot 2^{20}$$

$$s_e \cdot 2^{20} \cdot t_e \cdot 2^{20} = s_m \cdot t_m = r_m = r_e \cdot 2^{40}$$

It may then be necessary to "scale down" r_m so that its machine representation does not exceed the capacity of a 36 bit word. Thus the choice of scaling for a number depends not only on the number of fractional bits to be represented but also on the machine restriction of 36 bits per word. In general the coder must consider three things in scaling.

(a) All numbers must be scaled so that the "machine representations" of these numbers are integers.

(b) The routine should be coded so that the scaling is always known.

(c) Frequent rescaling may be necessary in the routine if the values of the numbers are not to exceed the capacity of a computer word.

In many computations involving scaled numbers, the desire to retain the maximum amount of "precision", i.e., maximum number of significant bits, create a problem. To solve this problem, the numbers that enter into the calculations, as well as the numbers resulting from these calculations, are "normalized". A normalized number is a number which has been scaled up in a 36-bit register so that the left-most stage of the register contains the sign bit of the number with the most significant bit of the number contained in the next adjacent stage, i.e., the number n is $2^{35} > n \geq 2^{34}$.

The following are examples of normalized numbers:

30 00000 00000 is 00 00000 00003, (+3) normalized,

47 77777 77777 is 77 77777 77774, (-3) normalized.

Numbers are normalized easily in the Univac Scientific by using the Scale Factor (SFuv) instruction. This instruction normalizes the number $(u)_i$ in A_R and stores the number indicating the left shifts necessary to restore the number $(u)_f$ to its original state. This shift count k is stored in the <u>v portion</u> of the register whose address is given by the v of the Scale Factor instruction.

In case the u address of the Scale Factor instruction is the Accumulator the relationship between the initial and final contents of the Accumulator is as follows:

$(A)_f = (A)_i \cdot 2^s$ where s is the scale factor.

The relationship between s and k is

$s = -k$ if $0 \leq k \leq 36$

$s = 72-k$ if $37 \leq k \leq 71$

If $k = 0$, $(A)_i$ was properly positioned before shifting.

If $k = 37$, $(A)_i$ is all ones or zeroes.

To illustrate the use of the Scale Factor instruction, consider the problem of normalizing the product of two numbers x_m and y_m where these numbers are themselves scaled machine copies as follows:

$$x_m = x \cdot 2^r$$

$$y_m = y \cdot 2^s.$$

The routine would be as follows:

```
C1        MP        v1        v2        form product

C2        SF        A         t1        normalize (A)

C3        TP        A         t2        place normalized product in t2
```

C4	TP	t1	A	place k in A
C5	TJ	t3	C7	is k < 37?
C6	RA	v3	t4	add 72_{10} to (v3)
C7	RS	v3	t1	subtract k from (v3)
C10	--	--	--	next instruction
t1			0	k, (zero initially)
t2			0	normalized product
t3			45	constant, 37_{10}
t4			110	constant, 72_{10}
v1	--	--	--	x_m
v2	--	--	--	y_m
v3	--	--	--	r + s initially.

The product formed by the first instruction is

$$(v_1) \cdot (v_2) = x_m \cdot y_m = x \cdot y \cdot 2^{r+s}.$$

The remaining instructions form the product

$$(t_2) = x \cdot y \cdot 2^{(v3)f}$$

where $(v_3)_f$ is

$$r + s - k \text{ if } k < 37$$

$$r + s + 72-k \text{ if } k \geq 37.$$

An overall relationship would be expressed as follows:

$$x \cdot y = (v_1) \cdot (v_2) \cdot 2^{-(v_3)_i} = (t_2) \cdot 2^{-(v_3)_f}.$$

It should be noted that the 21 leftmost stages of t1 must initially contain zero since the Scale Factor instruction writes only into the lower order 15 stages of t1.

If a routine is coded such that only the mantissas of scaled numbers are stored in the computer and operated upon by the program, the routine is said to be coded in <u>fixed point</u>. In this case the programmer must himself keep an account of the characteristics associated with each mantissa at every step of the computation. The programmer must also be aware of the size of the mantissas and scale them down when there is danger of exceeding the capacity of a computer word. On the other hand, if the mantissas are carried in normalized form, the result of every arithmetic operation must be normalized.

The programmer must be exceedingly careful in the record which he keeps of the characteristics. A mistake may result in his having to rescale the problem from that point on. This accounting is generally listed on the coding sheets. The characteristic is recorded beside each instruction which changes the scaling of a mantissa. Great care must also be exercised in judging the magnitude of the mantissas. If unanticipated overflow occurs, the problem may have to be rescaled.

It is evident that the fixed point coding of a problem can be complex; consequently it is sometimes preferable to store in the computer the characteristics as well as the mantissas and use subroutines which, before each arithmetic operation, effect the scaling of the mantissas on the basis of their associated characteristics. The result of each arithmetic operation is then stored as a mantissa with associated characteristic. This manner of coding is known as <u>floating point</u> coding. Various schemes may be used for the storage of the mantissa and characteristic of a floating point number. If the mantissa and characteristic are stored at separate locations, the floating point number is said to be "unpacked"; if the mantissa and characteristic are stored together at one location, the number is said to be in "packed" form. A common floating point packed form for the Univac Scientific allows 28 bits for the mantissa and eight bits for the characteristic. Generally the mantissa is normalized although this is not necessary. Should one wish to retain an indication of the number of significant bits in the results, the mantissas may be carried unnormalized.

As an example of one floating point packed form which has been used on the Univac Scientific, consider the following representation of the number s by the components s_1 and s_2 of 28 bits and eight bits, respectively.

$$\text{If } s \neq 0 \quad 1 > |s_1| \geq 1/2, \; 256 > s_2 + 128 \geq 0$$

$$\text{If } s = 0 \quad s_1 = s_2 + 128 = 0$$

The number s is represented in the computer by $s_1 \cdot 2^{35}$ and $s_2 + 128$, the machine forms of its mantissa and characteristic. Note that the mantissa is normalized. When the characteristic is represented in this way, as $s_2 + 128$, it is said to be a "biased" characteristic. (The characteristic might also be represented by eight bits as a "signed" characteristic, $128 \geq s_2 \geq -128$.) The packed representation of the number s is positioned at a location as follows:

28 bits	8 bits
normalized mantissa	biased characteristic

The choice of a representation for floating point numbers is influenced by many considerations which may vary from installation to installation.

Elaborate systems of subroutines have been developed to perform arithmetic operations with floating point numbers. The price which the programmer pays for relief from the complexities of fixed point scaling is increased execution time for the arithmetic operations. For instance, with a floating point system which uses a packed representation, the execution times for the operations of addition, multiplication, and division may be on the order of 60 times that of the corresponding machine operations. Floating point operations with unpacked operands are considerably faster. Here the multiples of the increase in time over the computer operations are approximately 20, 3, and 3 for addition, multiplication, and division, respectively. In general, floating point coding is used to reduce the elapsed time between receipt of a problem and the production of answers. Floating point notation is particularly useful in problems in which the numbers involved vary widely and where only crude predictions can be made of the amount of variation.

d. MULTIPLE PRECISION. - In addition to scaling in order to modify the range of numbers which may be represented in the computer, it is possible to extend the range by representing numbers in a <u>double precision</u>, <u>triple precision</u> or, in general, <u>n-precision</u> form. In this case two, three, or n computer words are used for the storage of a single number. For example, two locations are reserved in storage to represent a double precision number. The representation may be (A_L) and (A_R) when the number is in the Accumulator. The addition of two numbers so represented is easy and fast using the Split instructions. In the case of multiplication or division, an end correction must be provided for each word whose sign bit is "1". Another method is to select a positive constant k and represent a double precision number N as

$$N = q \cdot k + r$$

with a side condition on r to require unicity. Both q and r are single precision numbers. For example, let $k = 2^{34}$ with $0 \leq r < 2^{34}$. With this representation addition is slower but multiplication and division are markedly easier.

Similar schemes may be used for n-precision computing. Again, a number of considerations enter in the choice of a representation of multiple precision numbers.

e. CHOICE OF NUMBER NOTATION. - In choosing the number representation to be used in solving a given problem on the computer, the accuracy which is required is first considered. If a large number of significant bits is demanded, a multiple precision representation may be required. Secondly, the choice between a fixed point and a floating point notation is made on the basis of running time on the computer versus programming time. Floating point routines are frequently used to simplify the coding of "one shot" programs (the program is to be used once) where the emphasis is on elapsed time from problem formulation to solution. Generally if a program is to be run repeatedly, it is coded in fixed point in order to minimize the running time.

4. NOTES ON THE INSTRUCTIONS IN THE UNIVAC SCIENTIFIC REPERTOIRE.

In the following paragraphs, pecularities, tricks, and pitfalls which confront a programmer are discussed. In almost all cases the unusual circumstances which must be noted are due to particular features of the computer logic. Certain of these features create problems which are not immediately obvious.

a. OPERATIONS INVOLVING THE ACCUMULATOR.

(1) NEGATIVE ZERO. - Since the basic arithmetic operation of the Univac Scientific is subtraction and the "1's" complement number system is used, negative zero (72 "ones" in the Accumulator) cannot be generated by any series of arithmetic operations. Consider the operations performed by the instructions below:

```
C1    RA   e1        e2        form sum
C2    RS   A         e3        form difference
.
.
.
e1    77 77777 77776           -1 in decimal
e2    00 00000 00001           +1 in decimal
e3    77 77777 77777           -0 in decimal.
```

Using, for exemplary purposes, single length registers of four bits and an Accumulator of eight bits, the operations can be shown as follows:

```
Contents of
Accumulator        Operations

0000 0000   clear A                                 ┐
0000 0001   "add D(e1)" by subtracting D(e1)'       │
---------                                           │
1111 1111                                           │   Operations of
        1   end around borrow                       ├   RA at C1
---------                                           │
1111 1110                                           │
1111 1110   "add D(e2)" by subtracting D(e2)'       │
---------                                           ┘
0000 0000   result of executing (C1)                ┐
0000 0000   transmit (A_R) to X, then clear (A)     │
1111 1111   "add D(X)" by subtracting D(X)'         │
---------                                           │
0000 0001                                           │   Operations of
        1   end around borrow                       ├   RS at C2.
---------                                           │
0000 0000                                           │
1111 1111   subtract D (e3)                         │
---------                                           │
0000 0001                                           │
        1   end around borrow                       │
---------                                           │
0000 0000   result after executing (C2)             ┘
```

Continuing the above example, negative zero may be formed in the Accumulator using one of the "logical" instructions:

```
C3    SS    e3    0     form - (2^36 -1) in A
C4    CC    A     e3    form negative zero in A
```

i.e.,

```
0000 0000   result after executing (C2)             ┐
0000 1111   form (A) - S(e3)                        │
---------                                           │   Operations of
1111 0001                                           ├   SS at C3
        1   end around borrow                       │
---------                                           │
1111 0000   result of executing (C3)                ┘
```

1111 0000	transmit (A_R) to X, then clear (A_R)	Operations of CC at C4 .
1111 0000	form logical sum of (X) and (A_R)	
1111 1111	form logical sum of (A_R) and (e_3)	
1111 1111	result after executing (C4)	

Negative zero can be generated in the Accumulator by other series of instructions concluded by the Controlled Complement instruction (CCuv). Note that in the above example the content of e3 (normally considered a negative zero, mod 2^{36}-1) was considered as 2^{36}-1 (mod 2^{72}-1). Such is the case when any of the "split" or "logical" instructions are used.

If the above routine is continued with the execution of the following instruction, note that the content of the Accumulator (all "ones") must be considered as zero.

C5	LA	A	k	shift (A) left k places

i.e.,

1111 1111	result after executing (C4)	
1111 1111	do not clear (A), but clear (X)	Operations of LA at C5
1111 1111	"and D(X)" to A by subtracting D(X)' from A	
0000 0000	(A) is now all zeros	
0000 0000	shift (A) left k places	
0000 0000	result after executing (C5)	

Thus, if the content of the Accumulator is all "one's", its content must be considered to be all zeros except in the following cases.

If either an Equality Jump or Threshold Jump instruction is executed with the content of the Accumulator all one's, the result of testing (A) is as follows.

If (u) is 2^{36}-1 (negative zero),

EJuv shows an equality, takes (v) as NI
TJuv results in a positive (A), continues present sequence

If (u) is all zeros (positive zero),

EJuv does not show an equality, continues sequence
TJuv results in a negative (A), takes (v) as NI

In none of these cases will the content of A be restored to negative zero. The Accumulator will be set to all zeros after the execution of any of these instructions.

If a Zero Jump (ZJuv) is executed when the content of the Accumulator is all one's, (A) tests as not being equal to zero, but as the result of executing the ZJ the Accumulator is set to all zeros.

(2) SINGLE OR DOUBLE LENGTH EXTENSIONS. - A common error involving transmissions to the Accumulator and a subsequent testing of its contents is illustrated by the following example:

```
C1    SP    e1      0             enter variable in A
C2    EJ    e2      -             compare (A) and (e2)
C3    --    --      -
.
.
.
e1    40    00000 00000           variable (here shown as a constant)
e2    40    00000 00000           a constant.
```

After (C1) is executed, (A) is 00 00000 00000 40 00000 00000. The EJ at C2 tests to determine if (A) and D (e2) are equal which is not true in this case since D(e2) = 77 77777 77777 40 00000 00000. A correct method of testing for the equality of these two values, (e1) and (e2), could be coded as follows:

```
C1    TP    e1      a
C2    EJ    e2      -
C3    --    --      -
.
.
.
```

By using the Transmit Positive instruction, the double length extension of (e1) is formed in the Accumulator.

(3) COMMON PITFALLS. - Many instructions make use of the Accumulator without explicitly referencing it as one of the execution addresses.

It should be remembered that the initial contents of the Accumulator may be destroyed when executing certain of these instructions which use the Accumulator. In certain cases this may lead to erroneous results when the Accumulator is used as the u or v address of the instruction. A study of the section Sequential Listing of Instructions will indicate the peculiarities arising from such situations. The volume entitled <u>Content of Registers</u> presents in tabular form the results of such peculiarities.

Similar peculiarities result from addressing the Q Register as u or v of an instruction.

b. SHIFT INSTRUCTIONS. - All shifts performed are <u>left end-around</u> shifts. When a shift is performed the left-most portion of the number shifted in the Accumulator (or Q register) becomes the right-most portion of the number. It should be remembered that a left shift of "k" places is equivalent to a <u>modular</u> multiplication by 2^k. Depending upon the significance attached to the sign-bit, the modular value of the shifted number will be in the range of the <u>signed</u> numbers possible to represent in the register or in the range of the system of binary numbers directly represented in the register. If a significant "1" (or "0" if the number has a negative representation) is shifted into the sign-bit position of the register, the "sign" of the shifted number must be regarded as opposite in sign to the initial number unless the shifted number is referenced for later use by one of the Split instructions.

The mathematical statement expressing a left shift of (u) by k is

$$(u)\cdot 2^k - \left[(u)\cdot 2^{k-m}\right] 2^m + (u)\cdot 2^{k-m}$$

where the brackets mean the greatest integer contained in the register and

$m = 72$ for the Accumulator where $k \leq 72$
$m = 36$ for the Q Register where $k \leq 36$.

An equivalent right shift can be obtained from a left shift by observing that a right shift of k places = m-k left shifts.

When the v address portions of the Left Shift in A, LAuk, and Left Shift in Q, LQuk, instructions contain ones in other than positions $v_6 \ldots v_0$, transmission back to the u address of the shifted quantity contained in A or Q can be stopped. The peculiarities resulting from such a condition are discussed in the section Sequential Listing of Instructions.

This property of these shift instructions is useful whenever it is desired to leave the original content of u undisturbed. For example, consider the instruction LA 01000 32002.

The operations resulting from this instruction are:

Clear A

D (01000) $\longrightarrow$ A

(A) is shifted two places to the left

The shifted result is <u>not</u> transmitted to 01000 but left in A.

An operation possible for some u addressed RAS locations is:

Transmit (positive) to A and Q the content of u after (u) has been shifted k places to the left; e.g.,

LQ 01100 32005 or also LA 00100 31005

The following operations result from these instructions:

$(00100) \longrightarrow Q$ or $(00100) \longrightarrow A$

(Q) or (A) is shifted 5 places to the left

$D\ (Q) \longrightarrow A$ or $(A_R) \longrightarrow Q$

A short summary showing the results of coding the 54uk and 55uk instructions for transmissions to A or Q of $\dot{u}$ (= RAS) shifted is given below.

u address	v address	Shifted (RAS) transmitted
00000-00777	30----	(Fault, illegal address)
01000-01777	30----	to Q
02000-07777	30----	to A
00000-01777	31----	to Q
02000-07777	31----	to A
00000-07777	32----	to A

If the u address is an MD address when using this property of the Shift instructions (A) or (Q) is returned to the drum but not to the original address; e.g.,

LA 41234 32011

resulting in

Clear A

$D(\underline{4}1234) \longrightarrow A$

(A) is shifted 9 places to the left

$(A_R) \longrightarrow \underline{7}3234.$

c. "ROUND OFF" AND "SCALE DOWN" OPERATIONS. - During the course of many arithmetic operations it becomes necessary to change the scaling of a number. A simple method of scaling a number down would be to perform a left shift of k places equivalent to the desired right shift. For example, the instructions below multiply two numbers, (u) and (v), and scale the resulting product down five places.

MP	u	v	form product (u) · (v)
LA	A	00103	right shift 5 or (72-k) places
TP	A	A	eliminate unwanted bits.

The left shift instruction LA A 00103 (where $k = 103_8 = 67_{10} = 72-5$) performs an end-around left shift of the product 67 places. This is equivalent to an end-around right shift of five places. This shifting operation is equivalent to a division by 2^5 if the Transmit Positive instruction is included and if it is assumed that the multiplication of (u) and (v) did not generate more than 40 significant bits. The instruction TP A A performs the operation of clearing the least significant bits which appear in (A_L) after the shift operation. If these least significant bits were left in (A_L) and further arithmetic operations were performed on the shifted product, they would no longer be considered as least significant bits but as the most significant bits of the contents of the Accumulator. In fact, the bits that were shifted into (A_L) could also effect the sign of the Accumulator in any further arithmetic operations.

In the above example the product was shifted down without any consideration to the arithmetic error (truncation or round-off error) that the process may generate. Usually, before a "shift down" operation is performed in arithmetic problems, a "round-off" step is first programmed. In the above example the maximum error generated would be plus or minus one in the least significant place. The following method will minimize the round-off error to be in the range of plus or minus one-half in the least significant place. For simplicity let us assume that (u) and (v) are positive; then, to round-off and scale down, the instructions could be coded as

```
C1          MP    u      v       form product (u) · (v)
C2          SA    dl  00103      round-off and scale down
C3          TP    A      A       eliminate unwanted bits
 .
 .
 .
dl                      20       binary 10000.
```

In this example the product was scaled down five places after it was "rounded off" at the sixth place. The constant added would increase the number to be shifted down by one if the fifth bit of the number has "1". The general rule for rounding-off and scaling down by k places is to add 2^{k-1} (or subtract 2^{k-1} if the number to be rounded is negative) before the scaling operation occurs.

In "scale up" operations (multiplication by 2^k), round-off operations are not necessary, but care must be taken not to scale a number up beyond 71 bits if the number is in the Accumulator, and 35 bits if the number is in the Q register.

d. ACCUMULATIVE OVERFLOW. - The possibility of an overflow beyond the stage A_{71} of the Accumulator during repeated executions of a Multiply Add instruction is automatically checked by the computer, but often it is desirable to know if an overflow occurs into A_{35} and beyond during repeated executions of Multiply Add or Add and Transmit operations. An overflow into A_{35} can be determined by executing the Equality Jump instruction with its u address referencing the Accumulator; e.g., EJ A v. The equality of $D(A_R)$ and (A) is tested by this instruction; if there has been an overflow into A_{35}, $D(A_R) \neq (A)$. For example, if $(A)_i = 0000\ 1011$, $D(A_R) = 1111\ 1011$, indicating an overflow to or beyond A_{35}.

e. DIVIDE OVERFLOW. - The Divide instruction (DVuv) divides the 72-bit integer in the Addumulator by the 36-bit integer at address u, placing the quotient in the Q register and at address v, and leaving in A a positive remainder. Therefore, care must be exercised to insure that the quotient does not exceed 36 bits in length (35 bits plus a sign bit). If the quotient should exceed 36 bits, the computer will stop, giving a divide overflow indication on the control panel. The divide operation, as performed in the Univac Scientific, is defined as

$$(A)_i = (Q) \cdot (u) + R \text{ where } 0 \leq R < |u.|$$

It should be remembered that in the above definition, (Q), R, (u), and (A) are all integers.

Also to be noted is the following condition taken into account by the Divide instruction. Consider the definition for all divide cases when (A) is negative and when R, the remainder, is not zero. In order that R be positive, the magnitude of the quotient, (Q), must be equal to the magnitude of the "true quotient" plus one. The true quotient is the quotient as derived from the expression

$$|(A)| = (Q) \cdot (u) + R.$$

To obtain the true <u>rounded</u> quotient after a division, for all cases of the Divide instruction, the following expression may be used:

$$(A) + 1/2 \quad |(u)| = (Q) \cdot (u) + R,$$

and, assuming dl contains the dividend

```
C1      TM    u    A      form |(u)|, in A
C2      LA    A    107    form 1/2 |(u)| in A
C3      RA    A    dl     form (dl) + 1/2 |(u)|  in A
C4      DV    u    q      form rounded division.
```

It should be noted that the instruction at C2 puts an unwanted bit into A_{71}, but by using the Replace Add instruction, which, as addressed, only uses 36-bits of (A) to perform the addition, the bit in the A_{71} position is cleared out.

Since the Divide instruction produces as its result an <u>integer</u> quotient and remainder, the loss of significant bits may be troublesome in "fixed" or "floating point" operations if a scaling operation is not performed before a division. To illustrate how to perform this scaling operation, assume that the division b/c is desired where

$$1 < |b| \quad 1/2$$

and

$$1 < |c| \quad 1/2$$

and assume that b and c are scaled up in the registers el and e2 as

$$(el) = b \cdot 2^{35}$$

and

$$(e2) = c \cdot 2^{35}$$

or $b = b_m \cdot 2^{-35}$
$c = c_m \cdot 2^{-35}$

where b_m and c_m are the machine copies (integers) of the external true fractional numbers. The instructions would be coded as follows:

f1	LA	e1	20043	Scale b up 35 places and leave the result in A.
f2	DV	e2	Q	Divide by c and put the result in Q.

The Q register now contains the quotient scaled up 35 places; i.e.,

$$2^{35} > |q_m| \geq 2^{34} \text{ where } q_m = (Q)$$

or

$$1 > |q| \geq 1/2.$$

In the above example it was tacitly assumed that the magnitude of b was greater than the magnitude of c. If this were not the case, the result of executing (f2) would cause a divide overflow since

$$2^{36} > |q_m| \geq 2^{35}.$$

f. REPEAT INSTRUCTION. - Because of the complexity of the Repeat instruction, the details of the repeat operation are restated below.

(1) FORMAT OF REPEAT INSTRUCTION. - The Repeat instruction has the form RPjnw. The normal values of j are 0 through 3 and determine the advance of the execution addresses of the repeated instruction as follows.

if j=0, neither the u nor v address portion of the repeated instruction is advanced.

if j=1, the v address of the repeated instruction is advanced by one after each execution.

if j=2, the u address of the repeated instruction is advanced by one after each execution.

if j=3, both the u and v addresses of the repeated instruction are advanced by one after each execution.

The advance of the addresses occurs only in the Program Control Register (PCR) where the repeated instruction is held during the repeat operations. Therefore, the original form of the repeated instruction in storage is unaltered.

The value n determines the number of times the repeated instruction is to be executed. This value can vary through the range 0 through 4095 (decimal). If n is zero, the repeat operations are terminated without execution of the "repeated" instruction.

The "repeat termination address", w, replaces the v address portion of fixed address F_1 (00000 or 40001) during the execution of the RP instruction. (Normally, F_1 contains an MJjv instruction with a j of 0.) This feature provides the means of "jumping out" of the repeated operations after n executions of the repeated instruction.

(2) REPEATED NON-JUMP INSTRUCTION. - The repeated instruction is held in the Program Control Registers (PCR) and is executed n times. After the n^{th} execution, a jump to F_1 occurs. Because this instruction is usually an MJjv with a j of zero and a v of w, it produces a jump to the address originally specified by w of the RP instruction.

(3) REPEATED JUMP INSTRUCTION. - If the repeated instruction is a Threshold Jump (TJ) or Equality Jump (EJ), the jump to w may not occur. The repeated instruction is held in PCR and is executed not more than n times. If n executions occur with no jump condition being fulfilled, the normal termination, consisting of a jump to F_1, followed by a jump to w, occurs. However, if any execution of the repeated instruction fulfills a jump condition, the value j, n-r is stored in the v portion of Q (where j = j of RP instruction; and n-r = n of RP instruction minus r, the number of executions performed), and a jump is made to the address specified by v of the TJ or EJ instruction.

Similar circumstances are created by the Index Jump and Manually Selective Jump instructions with the exception that no indication of the number of executions is given by a storage in Q.

Other instructions (Interpret, Return Jump, Q Jump, Sign Jump, Zero Jump, Manually Selective Stop, and Program Stop) produce a jump or stop on the first execution, and thus behave as if no RP instruction preceded them.

(4) EXAMPLES USING THE REPEAT INSTRUCTION. - In many problems which are programmed for computer execution, it is important that the most efficient use of operating time and storage capacity be made. The Repeat instruction, an unusual logical feature of the Univac Scientific, makes it possible to realize a sizeable reduction in the time spent referring to storage for instructions and in the storage space devoted to holding instructions. By using the RP instruction, long sequences of operations can be governed by only two or three instructional references to storage. This is evidenced by the following examples.

Consider the accumulation of products, such as,

$$S = a_1b_1 + a_2b_2 + \ldots + a_nb_n = \sum_{i=1}^{n} a_ib_i$$

The Multiply Add instruction (MAuv) causes the product of the numbers located in storage with the addresses u and v to be added to the number already in the Accumulator, leaving the total result in the 72-bit Accumulator. If the MA instruction is repeated as shown below, the scalar product of two n-vectors is generated.

d1	RS	A	A	Clear Accumulator to zero
d2	RP	3, n	d4	Form sum of products
d3	MA	u_1	v_1	S = (u1) (v1) + (u2) (v2) + ... + (un)(vn)
d4	--	--	--	Next instruction
u1			a_1	
.			.	
.			.	Vector a
.			.	
un			a_n	
v1			b_1	
.			.	
.			.	Vector b
.			.	
vn			b_n	

Note the instruction at d1 which clears the Accumulator to insure that its initial content is zero.

Care should be taken that the contents of F1 are not inadvertently altered during a block transfer of n words to Rapid Access Storage. This pitfall is illustrated by the following example:

```
RP   30400       w
TP   50000   07600
```

The sequence of events which occurs during the execution of these instructions is as follows:

(a) Replace the lower order 15 bits of (F_1) with the address w

(b) Proceed to transmit the contents of

```
50000 to 07600
50001 to 07601
  .        .
  .        .
  .        .
50177 to 07777
50200 to 00000
  .        .
  .        .
  .        .
50377 to 00177
```

(c) Extract the next instruction to be executed from address F_1. Note that the contents of the address 00000, (F_1), have been replaced by the contents of address 50200 during the repeated transmissive operations. Because of this, the desired jump to wo no longer exists at this address.

The following example illustrates further the care that must be exercised when using the Repeat instruction.

```
00100    RP    30200    00303
00101    TP    45000    00200
```

The operations performed by this pair of instructions are:

(a) Replace the lower order 15 bits of (F_1) with the address 00303

(b) Proceed to transmit the contents of

```
45000 to 00200
45001 to 00201
  .        .
  .        .
  .        .
45102 to 00302
45103 to 00303
  .        .
  .        .
  .        .
45177 to 00377
```

(c) Extract the next instruction to be executed from address F_1. Note that in this example the original contents of 00303 have been replaced by the contents of address 45103. Therefore, the next instruction to be executed after the jump to 00303 is the instruction which was originally stored at 45103.

(5) The initial operations of the RP instruction replace the lowest order 15 bits of (F_1), the contents of the address 00000 (or 40001), with the address w of the RPjnw instruction. The terminating operations of the repeat sequence extract the next instruction to be executed from the address F_1

(unless the repeated instruction called for a jump to v). The instruction contained in F_1 is normally a jump instruction (usually an MJjv) referencing its v address; the address w written in by the RP. However, if the instruction contained in F_1 is not a jump instruction, or if it is a conditional jump (IJ, MJ, TJ or EJ) where the jump condition is not fulfilled, the next instruction to be executed after the execution of the instruction at F_1 is taken from one of the following addresses:

If j was 0, NI from address 40000
If j was 1, NI from address 70000
If j was 2, NI from address 60000
If j was 3, NI from address 50000.

These are the addresses that are left in the Program Address Counter (PAK) after the RP instruction, because of the use of PAK as the counter for the number of executions to be performed on the repeated instruction.

(7) PRINT AND PUNCH INSTRUCTION, PR-v AND PUjv. - Because of the operands of the Print and Punch instructions requiring only a six bit storage space, these instructions can be coded so that a savings of storage space is effected. The v address of PUjv or PR-v may reference any instruction whose v address references the Accumulator or Q Register, or whose v address portion is not used in the instruction, e.g., AMjn-, BMjn-, or PS--. For example, note the following instructions:

```
01001    PR    00000    01050    Print the number "1"
  .              .
  .              .
  .              .
01050    MP    00010    31052
```

Note that only one character is printed or punched with each output reference. In order to print or punch out in octal a 36 bit word, twelve Print or Punch instructions would be needed.

To illustrate the use of these instructions, consider the following routine which prints out on the typewriter one word in octal.

```
C1    PR    0     e10     Print carriage return

C2    LQ    d1    3       Shift word to be printed 3 places
                          to the left

C3    QT    d2    A       Mask out 3 bits or one octal digit.

C4    AT    d3    C5      Compute print instruction which is NI.

C5    PR    0     ei      Print one octal digit.

C6    IJ    d4    C2      Test for end, if not finished go to C2

C7    PS    --    --      End.
```

```
d1     xx    xxxxx  xxxxx    Number to be printed

d2     00    00000  00007    Mask

d3     PR      0     e0      Dummy command

d4     00    00000  00013    Counter, n-1 = 11₁₀

e0     00    00000  00037  ┐ (0)
e1     00    00000  00052  │ (1)
e2     00    00000  00074  │ (2)
                           │          Flexowriter codes for
e3     00    00000  00070  ┤ (3)      the digits 0 through 7.
e4     00    00000  00064  │ (4)
e5     00    00000  00062  │ (5)
e6     00    00000  00066  │ (6)
e7     00    00000  00072  ┘ (7)

e10    00    00000  00045             Flexowriter code for
                                      carriage return.
```

APPENDIX A

NUMBER SYSTEMS

NUMBER SYSTEMS

1. GENERAL

Any positive integer N can be expressed in the form

$$N = A_n r^n + A_{n-1} r^{n-1} + \cdots + A_1 r^1 + A_0 r^0$$

where $0 \leq A_i < r$ and r is any integer greater than 1. The integer r is called the base or radix of the number system. The only radix values that need to be considered for programming for the UNIVAC SCIENTIFIC are 10, 8 and 2. The number systems with these radices are called decimal, octal, and binary systems, respectively.

An integer N expressed as a number in the usual decimal notation implies that the number is a refinement of a polynomial in powers of 10 with coefficients which satisfy the relation $0 \leq A_i < 10$. For example, N = 308 implies the polynomial $N = 3 \cdot 10^2 + 0 \cdot 10^1 + 8 \cdot 10^0$. Accordingly, an integer N = 1011 in the binary system with $0 \leq A_i < 2$ may be expressed as the polynomial $N = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$ (= 11 decimal); and an integer N = 126 in the octal system with $0 \leq A_i < 8$ may be expressed as the polynomial $N = 1 \cdot 8^2 + 2 \cdot 8^1 + 6 \cdot 8^0$ (= 86 decimal).

Because electronic and magnetic circuits consist primarily of bi-stable elements, the computer uses the binary representation in all internal manipulations of numbers. As will be seen later, there is a very simple relationship between the binary and octal representations of a number. Since conversion between these two systems is almost immediate and since the octal notation is shorter, programs are encoded with octal notation representing binary numbers. It is desirable for people who prepare programs for the computer to be familiar with methods of converting numbers from decimal to binary to octal form and vice versa. When the base of a number is not apparent from the context, it will be specified by a subscript as shown in the following example:

$$34_{10} = 42_8 = 100010_2$$

2. CHANGE OF BASE

Since the manual conversion of numbers from one system to another involves the arithmetic operations of addition and multiplication, it is necessary to know the sums and products of certain pairs of integers expressed in the octal and binary systems. In doing arithmetic operations by hand, the ability to remember the sums and products of pairs of decimal digits is utilized, although these sums and products are tabulated in decimal addition and multiplication tables. Similar tables presented below are necessary for arithmetic operations with octal and binary digits.

OCTAL ADDITION TABLE

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

OCTAL MULTIPLICATION TABLE

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

The addition table and the multiplication table for binary numbers are shown below

BINARY ADDITION TABLE

	0	1
0	0	1
1	1	10

BINARY MULTIPLICATION TABLE

	0	1
0	0	0
1	0	1

a. CONVERSION OF INTEGERS. - The simplest way of changing a binary or octal integer to its decimal equivalent is to expand it to, and evaluate the sum of the terms of, the polynomial expression given on a previous page. The conversion from decimal to octal or binary presents more of a problem. Two methods are given which can be used for these conversions. The first of these is as follows:

(1) Given two integers N and D, the latter being positive, it can be shown that there exist unique integers Q and R such that $N = QD + R$ where $0 \leq R < D$. This is, of course, true regardless of the choice of base used in expressing N. Hence, if N_{10} and N_8 are the decimal and octal expressions for the same integer and if D_{10} and D_8 are the decimal and octal expressions for another (positive) integer, we can compare the two equations

$$N_{10} = Q_{10} \cdot D_{10} + R_{10}, \ 0 \leq R_{10} < D_{10}$$
$$N_8 = Q_8 \cdot D_8 + R_8, \ 0 \leq R_8 < D_8$$

and conclude that $Q_{10} = Q_8$ and $R_{10} = R_8$. This fact is used in the process of converting an integer from base 10 to base 8 in the following way: The decimal expressions for an integer N expressed as N_{10} and N_8 are

$$N_{10} = d_n 10^n + d_{n-1} 10^{n-1} + \cdots + d_1 10 + d_0, \ 0 \leq d_i < 10$$
$$N_8 = e_m 8^m + e_{m-1} 8^{m-1} + \cdots + e_1 8 + e_0, \ 0 \leq e_i < 8$$

Dividing the polynomials by decimal 8 gives the same remainder in either case, but from the second expression, it is obvious that this remainder is e_0 while the quotient is $e_m 8^{m-1} + e_{m-1} 8^{m-2} \cdots + e_2 8 + e_1$. Dividing this quotient by 8 gives the new quotient $e_m 8^{m-2} + \cdots e_3 8 + e_2$ and the new remainder e_1. Continuing division results in the successive remainders $e_0, e_1, e_2, \cdots, e_m$ which are the digits of N_8 in reverse order. In a numerical case, the successive divisions of N_{10} by decimal 8 yield remainders which are actually the decimal equivalents of the digits e_0, e_1, ..., e_m.

To illustrate this method, the number expressed as 1492 in decimal notation is changed to its octal equivalent as follows:

Decimal Expression		Octal Equivalent
$8\,\overline{)1492}$,	Remainder 4	4
$8\,\overline{)186}$,	Remainder 2	2
$8\,\overline{)23}$,	Remainder 7	7
$8\,\overline{)2}$,	Remainder 2	2
0		

Thus $1492_{10} = 2724_8$

Conversion of an integer N from base 8 to base 10 can be carried out in a similar fashion by dividing N_8 by 12_8 (= 10_{10}) to yield the octal equivalents of the digits of N_{10}.

The conversion of an integer N from base 10 to base 2 or from base 2 to base 10 can be discussed according to the method outlined above with N_{10} and N_2 represented as follows:

$$N_{10} = d_n 10^n + \ldots + d_1 10 + d_0, \quad 0 \leq d_i < 10$$
$$N_2 = b_k 2^k + \ldots + b_1 2 + b_0, \quad 0 \leq b_i < 2$$

N_{10} would be divided by 2 to yield the decimal equivalents of the digits of N_2, and N_2 would be divided by 1010 (= 10_{10}) to yield the binary equivalents of the digits of N_{10}.

(2) The method that follows has the advantage that it yields the desired digits in normal order, but it has the disadvantage that it requires a knowledge of the values of powers of one base in terms of another. This method

requires the procedure below:

Given a number N_r to express as $N_b = a_nb^n + a_{n-1}b^{n-1} + \ldots + a_1b^1 + a_0b^0$, the coefficients $a_n \ldots a_0$, $0 \le a_i < b$, may be found as follows:

1. Determine the highest power n such that $b^{n+1} > N_r$. Then, let $N_r = N_n$. Divide N_n by b^n, yielding the quotient a_n and the remainder $N_{n-1} = a_{n-1}b^{n-1} + \ldots + a_0b^0$.
2. Repeat the divisions of N_i (i = n, n-1, ..., 0) by b^i until the last division yields the quotient a_0.
3. A quotient (or coefficient) a_i may be zero and must be represented in its proper order in the final result.

As an example of this method the number expressed decimally as 137 is changed to its octal equivalent as follows:

$$N_n = a_nb^n + a_{n-1}b^{n-1} + \ldots + a_0b^0 = N_b$$

$$r = 10,\ b = 8,\ N_r = N_n = 137,\ n = 2\ (8^3 > 137)$$

$$137 = 2 \cdot 64 + 9 \qquad a_2 = 2,\ N_1 = 9$$

$$9 = 1 \cdot 8 + 1 \qquad a_1 = 1,\ N_0 = 1$$

$$1 = 1.1 + 0 \qquad a_0 = 1$$

The coefficients derived above are the digits of N_8; thus $N_{10} = 137 = N_8 = 211$.

(3) Although it would be possible to use the above methods in converting an integer from base 8 to base 2 and vice versa, it is easier to convert by inspection. This can be done because of simple relationship, discussed subsequently, between N_8 and N_2, dependent upon the fact that $2^3 = 8$.

The equivalent polynomials for an integer N expressed as N_8 and N_2 are

$$N_8 = e_m8^m + e_{m-1}8^{m-1} + \ldots + e_28^2 + e_18^1 + e_08^0,\quad 0 \le e_i < 8$$

$$N_2 = b_k2^k + b_{k-1}2^{k-1} + \ldots + b_22^2 + b_12^1 + b_02^0,\quad 0 \le b_i < 2$$

The expression for N_2 may be factored as follows:

$$N_2 = \ldots + (b_{3m+2}2^2 + b_{3m+1}2^1 + b_{3m})2^{3m} + \ldots + (b_2 2^2 + b_1 2^1 + b_0)2^0$$

with the integral range $m = 0, 1, 2, \ldots, p/3,\ k \geq p \geq k-2$

The value of p which is a multiple of three is chosen. It is seen by comparing termwise the expressions for N_2 factored and N_8 that a common component exists, $(2^3)^m = 8^i$. Hence, the coefficients of each term are equal since the polynomial expressions themselves are equal. Thus the digits of N_8, given its binary equivalent, are as follows:

$$\begin{aligned} e_0 &= (b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0) \\ e_1 &= (b_5 . 2^2 + b_4 \cdot 2^1 + b_3) \\ e_2 &= (b_8 \cdot 2^2 + b_7 \cdot 2^1 + b_6) \\ &\cdot \quad\quad \cdot \\ &\cdot \quad\quad \cdot \\ &\cdot \quad\quad \cdot \\ e_m &= (b_{p+2}2^2 + b_{p+1}2^1 + b_p) \end{aligned}$$

Note that the condition $0 \leq e_i < 8$ is satisfied by the above relationship.

The simple procedure of the conversion from N_2 to N_8, according to the preceding method, is shown by the following example:

Given $N_2 = 1101001111$, the digits of its equivalent in octal notation, N_8, are derived by evaluating each group of three binary digits, beginning at the right. Thus, N_2 above yields $N_8 = 1517$. (The binary number may be expanded by assuming zeros at the left.)

To convert a number from octal to binary notation, each octal digit is replaced by its binary equivalent.

b. CONVERSION OF FRACTIONS. - Fractions in the binary and octal number systems are represented in a fashion similar to fractions in the decimal system.

For example, the number N, represented by the mixed decimal number 5.78125, is equal to the fraction $\frac{578125}{10^5}$. The number N in binary notation, represented by the mixed number 101.11001, is equal to the binary fraction $\frac{10111001}{100000}$. To represent a fraction, base r, whose denominator is equal to ar^p with $0 \leq a < r$, as a number with a radical point, place the radical point p places to the left of the right-most digit of the numerator.

Numbers represented by the fractions of one system may be translated to their representations in other systems by applying the methods described previously to both the integral numerator and denominator. For example, the fraction in the binary notation above may be converted to decimal notation as follows:

$$\frac{2^7 + 2^5 + 2^4 + 2^3 + 2^0}{2^5} = \frac{185}{32} = 5.78125$$

A fraction in the binary system may be changed to its octal notation by the grouping method described previously. Applying this method to the same fraction, base 2, yields the fractions, base 8, $\frac{271}{40} = \frac{27.1}{4} = 5.62$. This octal number may be expressed as the fraction $\frac{562}{100}$ which is equal to the decimal fraction $\frac{5 \cdot 8^2 + 6 \cdot 8 + 2}{8^2} = 5.78125$.

Fractions, expressed with a radical point, of one number system may be translated to numbers of another system by the method described subsequently. If the fraction is a mixed number, the integral portion is converted by one of the methods for integral conversion previously explained. The fractional conversion may be explained by establishing the polynomial expressions for a number F expressed as a fraction in two systems. The expansions of F_{10} and F_8 will be used for exemplary purposes. The expressions for a fraction F are as follows:

$$F_{10} = d_{-1}10^{-1} + d_{-2}10^{-2} + \ldots + d_{1-n}10^{1-n} + d_{-n}10^{-n},\ 0 \le d_i < 10$$

$$F_8 = e_{-1}8^{-1} + e_{-2}8^{-2} + \ldots + e_{1-m}8^{1-m} + e_{-m}8^{-m},\ 0 \le e_i < 8$$

Multiplication by 8 of the polynomials above, yields the expressions

$$F_8 \times 8 = e_{-1}8^{0} + e_{-2}8^{-1} + \ldots + e_{-m}8^{1-m}$$

and $F_{10} \times 8 = d_{-1}10^{-1}.8 + d_{-2}10^{-2}.8 + \ldots + d_{-n}10^{-n}\cdot 8$

The multiplication of $F_8 \times 8$ results in a mixed number with $e_{-1}8^0$ as the integral portion; therefore, the number to the left of the decimal point of $F_{10} \times 8$ must be the decimal equivalent of e_{-1}. This number may be a zero or an integer depending on the value of the first decimal digits d_{-1} of F_{10}. To find the decimal digit equivalent to the second octal digit e_{-2}, multiply by 8 the fractional portion of $F_{10} \times 8$. This multiplication will result in a number whose portion to the left of the decimal point is equivalent to the integral portion, $e_{-2}8^0$, of 8 times the fractional portion of $F_8 \times 8$. Thus, successive multiplications by 8 of the fractional portions of F_{10} yield the decimal equivalents of the octal digits e_{-1}, e_{-2}, ..., e_{-m}. As an example, to convert the fraction 5.78125, base 10, to its octal equivalent the procedure would be as follows:

	Decimal expression	Octal equivalent
	5.78125	5
.78125 x 8 =	6.25	6
.25 x 8	2.00	2

Thus $5.78125_{10} = 5.62_8$.

Similar conversions are made with other number systems by using the appropriate number as the multiplier. Note that a terminating fraction of one base need not lead to a terminating fraction of another base. For example, $0\cdot 10_{10} = .0631463146314\ \ldots_8$.

3. REPRESENTATION OF SIGNED NUMBERS

The modulus of a number system is the discrete number of quantities which can be represented by the system. The modulus of a number system to be represented by the elements of a computer is fixed by the design of the machine. This discussion will consider only the representation of a binary system by k parallel stages of bi-stable elements. Each stage represents a digit of N_2. The state of each stage (indicating a 0 or 1) represents a coefficient of a term of the polynomial

$$N_2 = b_{k-1}2^{k-1} + \ldots + b_1 2^1 + b_0 2^0, \; 0 \leq b_i < 2.$$

For exemplary purposes the value $k = 4$ stages will be used. Thus, if a computer had four bi-stable elements the modulus of the binary system which could be represented by the machine would be $2^4 = 16$, with the range of binary numbers from 0000 to 1111. It is desirable that this range of binary numbers from 0 to 2^k-1 represents both positive and negative values since it is not possible machine-wise to indicate negative numbers by a minus sign designation. Therefore, the binary digit represented by the stage 2^{k-1} is reserved to indicate the sign of a number, and negative numbers are represented machine-wise by a complement form. The simplest way to represent a negative number in such a manner in a computer with bi-stable elements is to use the one's complement of its absolute value. This entails subtracting each binary digit of the number from the quantity one or subtracting the binary representation of the absolute value of a negative number of a system of signed numbers, modulus 2^k, from the binary representation of 2^{k-1}. Thus, the process of forming the one's complement representation of the negative number -3, with k = 4, would be as follows:

```
       1111
minus  0011  binary representation of |-3 |
       1100  machine representation of -3 .
```

Note that forming the one's complement representation of a negative number can be accomplished machine-wise, if the absolute value is represented by k stages, by merely reversing the state of a bi-stable element: each representation of one is replaced by a zero representation and vice versa.

In a one's complement binary system the range of values of signed numbers that can be represented by k stages is $1-2^{k-1}$ to $2^{k-1}-1$ with the left-most stage indicating the sign of the number represented. The table below shows the correspondence of signed numbers to their representation in one's complement system with k = 4.

Signed Decimal Number	Signed Binary Number	One's Complement Binary Number
7	+111	0111
6	+110	0110
5	+101	0101
4	+100	0100
3	+011	0011
2	+010	0010
1	+001	0001
0	+000	0000
-0	-000	1111
-1	-001	1110
-2	-010	1101
-3	-011	1100
-4	-100	1011
-5	-101	1010
-6	-110	1001
-7	-111	1000

Note that the moduli of the preceding systems of signed numbers, as shown represented by a one's complement binary system, is reduced from 2^4 to 2^4-1 if the two zeros represented are considered a unique value. It is desirable in computer operations to have only one representation of zero, 00...00 or 11...11. In actual arithmetic operations by the Univac Scientific the occurrence of only one of these representations of zero is possible. This reduces the modulus of a one's complement system represented by k stages of 2^k-1. The fundamental arithmetic operation of the computer determines which of these two states will represent zero. The basic arithmetic operation of subtraction as performed in the UNIVAC SCIENTIFIC results in a zero representation, derived from any internal arithmetic operation, to be the simultaneous zero state of each stage. (Addition is performed by a subtractive process by subtracting the one's complement of the addend from the augend to form the difference, which will, in this case, be the sum.)

Thus the representation of a zero occurring when a number is subtracted from itself or when the absolute value of a negative number is added (by a subtractive process) to its negative value will be a series of zeros if these operations are performed by a subtractive process. The following examples illustrate this process with a four-stage one's complement binary number representation.

$$\begin{array}{rcrcrcrc} & 0001 & & 1 & & 1110 & & -1 \\ \text{minus} & \underline{0001} & \text{minus} & \underline{1} & \text{minus} & \underline{1110} & \text{plus} & \underline{|-1|} \\ & 0000 & & 0 & & 0000 & & 0 \end{array}$$

Remember that the computer does not interpret numbers represented binarily as having signed values. Arithmetic operations deal with a system of numbers assuming an arrangement of:

```
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
```

A subtraction in binary, such as 0010 minus 0100, is interpreted as subtracting four times the quantity of one from 0010. By referring to the above system and counting backward modularly, the value of 1110 is derived. However, this is not the one's complement representation of the desired value of minus two. An additional quantity of one must be subtracted to yield the desired value 1101.

This additional quantity of one must be subtracted whenever the subtraction operation "counts through" the negative representation of zero, 1111. With a binary system representing a number system modulus 2^k-1, this correction for the transition point is embodied as a borrow propagated from the left-most stage, 2^{k-1}. The correction is made by applying the borrow to the right-most stage. Machinewise, the borrow propagated from the left-most stage is actually applied to the right-most stage and is known as an end-around or circular borrow. Using the notation of the one's complement system, modulus 2^4-1, the subtraction of a value of four from a value of two proceeds as follows:

```
          0010            2
minus     0100    minus  +4
          ----           --
          1110
borrow       1
          ----
          1101           -2 .
```

The subtraction was enabled by applying the borrow, propagated from the 2^3 stage, to the 2^0 stage.

Another example is provided by the simple process of forming the sum of zero and one by a subtractive process. Thus, using the same notation

```
        0000            0
minus   1110     minus -1
        ----           --
        0010
borrow     1
        ----
        0001           +1
```

In this case the end-around borrow is continued past the right-most stage until it can be made. It may continue only as far as the stage where the conditions first necessitated the borrow.

APPENDIX B

TABLE OF POWERS OF TWO

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

APPENDIX C

DECIMAL TO OCTAL CONVERSION TABLE

DECIMAL TO OCTAL CONVERSION TABLE
(0-4096)

0 Through 199

		UNITS									
		0	1	2	3	4	5	6	7	8	9
00	0	0	1	2	3	4	5	6	7	10	11
	1	12	13	14	15	16	17	20	21	22	23
	2	24	25	26	27	30	31	32	33	34	35
	3	36	37	40	41	42	43	44	45	46	47
	4	50	51	52	53	54	55	56	57	60	61
TENS	5	62	63	64	65	66	67	70	71	72	73
	6	74	75	76	77	100	101	102	103	104	105
	7	106	107	110	111	112	113	114	115	116	117
	8	120	121	122	123	124	125	126	127	130	131
	9	132	133	134	135	136	137	140	141	142	143
01	0	144	145	146	147	150	151	152	153	154	155
	1	156	157	160	161	162	163	164	165	166	167
	2	170	171	172	173	174	175	176	177	200	201
	3	202	203	204	205	206	207	210	211	212	213
	4	214	215	216	217	220	221	222	223	224	225
TENS	5	226	227	230	231	232	233	234	235	236	237
	6	240	241	242	243	244	245	246	247	250	251
	7	252	253	254	255	256	257	260	261	262	263
	8	264	265	266	267	270	271	272	273	274	275
	9	276	277	300	301	302	303	304	305	306	307

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

200 Through 399

UNITS

		0	1	2	3	4	5	6	7	8	9
02	0	310	311	312	313	314	315	316	317	320	321
	1	322	323	324	325	326	327	330	331	332	333
	2	334	335	336	337	340	341	342	343	344	345
	3	346	347	350	351	352	353	354	355	356	357
TENS	4	360	361	362	363	364	365	366	367	370	371
	5	372	373	374	375	376	377	400	401	402	403
	6	404	405	406	407	410	411	412	413	414	415
	7	416	417	420	421	422	423	424	425	426	427
	8	430	431	432	433	434	435	436	437	440	441
	9	442	443	444	445	446	447	450	451	452	453
03	0	454	455	456	457	460	461	462	463	464	465
	1	466	467	470	471	472	473	474	475	476	477
	2	500	501	502	503	504	505	506	507	510	511
	3	512	513	514	515	516	517	520	521	522	523
TENS	4	524	525	526	527	530	531	532	533	534	535
	5	536	537	540	541	542	543	544	545	546	547
	6	550	551	552	553	554	555	556	557	560	561
	7	562	563	564	565	566	567	570	571	572	573
	8	574	575	576	577	600	601	602	603	604	605
	9	606	607	610	611	612	613	614	615	616	617

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

400 Through 599

UNITS

		0	1	2	3	4	5	6	7	8	9
04	0	620	621	622	623	624	625	626	627	630	631
	1	632	633	634	635	636	637	640	641	642	643
	2	644	645	646	647	650	651	652	653	654	655
	3	656	657	660	661	662	663	664	665	666	667
	4	670	671	672	673	674	675	676	677	700	701
TENS	5	702	703	704	705	706	707	710	711	712	713
	6	714	715	716	717	720	721	722	723	724	725
	7	726	727	730	731	732	733	734	735	736	737
	8	740	741	742	743	744	745	746	747	750	751
	9	752	753	754	755	756	757	760	761	762	763
05	0	764	765	766	767	770	771	772	773	774	775
	1	776	777	1000	1001	1002	1003	1004	1005	1006	1007
	2	1010	1011	1012	1013	1014	1015	1016	1017	1020	1021
	3	1022	1023	1024	1025	1026	1027	1030	1031	1032	1033
	4	1034	1035	1036	1037	1040	1041	1042	1043	1044	1045
TENS	5	1046	1047	1000	1051	1052	1053	1054	1055	1056	1057
	6	1060	1061	1062	1063	1064	1065	1066	1067	1070	1071
	7	1072	1073	1074	1075	1076	1077	1100	1101	1102	1103
	8	1104	1105	1106	1107	1110	1111	1112	1113	1114	1115
	9	1116	1117	1120	1121	1122	1123	1124	1125	1126	1127

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

600 Through 799

UNITS

		0	1	2	3	4	5	6	7	8	9
06	0	1130	1131	1132	1133	1134	1135	1136	1137	1140	1141
	1	1142	1143	1144	1145	1146	1147	1150	1151	1152	1153
	2	1154	1155	1156	1157	1160	1161	1162	1163	1164	1165
	3	1166	1167	1170	1171	1172	1173	1174	1175	1176	1177
	4	1200	1201	1202	1203	1204	1205	1206	1207	1210	1211
TENS	5	1212	1213	1214	1215	1216	1217	1220	1221	1222	1223
	6	1224	1225	1226	1227	1230	1231	1232	1233	1234	1235
	7	1236	1237	1240	1241	1242	1243	1244	1245	1246	1247
	8	1250	1251	1252	1253	1254	1255	1256	1257	1260	1261
	9	1262	1263	1264	1265	1266	1267	1270	1271	1272	1273
07	0	1274	1275	1276	1277	1300	1301	1302	1303	1304	1305
	1	1306	1307	1310	1311	1312	1313	1314	1315	1316	1317
	2	1320	1321	1322	1323	1324	1325	1326	1327	1330	1331
	3	1332	1333	1334	1335	1336	1337	1340	1341	1342	1343
	4	1344	1345	1346	1347	1350	1351	1352	1353	1354	1355
TENS	5	1356	1357	1360	1361	1362	1363	1364	1365	1366	1367
	6	1370	1371	1372	1373	1374	1375	1376	1377	1400	1401
	7	1402	1403	1404	1405	1406	1407	1410	1411	1412	1413
	8	1414	1415	1416	1417	1420	1421	1422	1423	1424	1425
	9	1426	1427	1430	1431	1432	1433	1434	1435	1436	1437

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

800 Through 999

		UNITS									
		0	1	2	3	4	5	6	7	8	9
08	0	1440	1441	1442	1443	1444	1445	1446	1447	1450	1451
	1	1452	1453	1454	1455	1456	1457	1460	1461	1462	1463
	2	1464	1465	1466	1467	1470	1471	1472	1473	1474	1475
	3	1476	1477	1500	1501	1502	1503	1504	1505	1506	1507
TENS	4	1510	1511	1512	1513	1514	1515	1516	1517	1520	1521
	5	1522	1523	1524	1525	1526	1527	1530	1531	1532	1533
	6	1534	1535	1536	1537	1540	1541	1542	1543	1544	1545
	7	1546	1547	1550	1551	1552	1553	1554	1555	1556	1557
	8	1560	1561	1562	1563	1564	1565	1566	1567	1570	1571
	9	1572	1573	1574	1575	1576	1577	1600	1601	1602	1603
09	0	1604	1605	1606	1607	1610	1611	1612	1613	1614	1615
	1	1616	1617	1620	1621	1622	1623	1624	1625	1626	1627
	2	1630	1631	1632	1633	1634	1635	1636	1637	1640	1641
	3	1642	1643	1644	1645	1646	1647	1650	1651	1652	1653
TENS	4	1654	1655	1656	1657	1660	1661	1662	1663	1664	1665
	5	1666	1667	1670	1671	1672	1673	1674	1675	1676	1677
	6	1700	1701	1702	1703	1704	1705	1706	1707	1710	1711
	7	1712	1713	1714	1715	1716	1717	1720	1721	1722	1723
	8	1724	1725	1726	1727	1730	1731	1732	1733	1734	1735
	9	1736	1737	1740	1741	1742	1743	1744	1745	1746	1447

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

1000 Through 1199

UNITS

		0	1	2	3	4	5	6	7	8	9
10	0	1750	1751	1752	1753	1754	1755	1756	1757	1760	1761
	1	1762	1763	1764	1765	1766	1767	1770	1771	1772	1733
	2	1774	1775	1776	1777	2000	2001	2002	2003	2004	2005
	3	2006	2007	2010	2011	2012	2013	2014	2015	2016	2017
	4	2020	2021	2022	2023	2024	2025	2026	2027	2030	2031
TENS	5	2032	2033	2034	2035	2036	2037	2040	2041	2042	2043
	6	2044	2045	2046	2047	2050	2051	2052	2053	2054	2055
	7	2056	2057	2060	2061	2062	2063	2064	2065	2066	2067
	8	2070	2071	2072	2073	2074	2075	2076	2077	2100	2101
	9	2102	2103	2104	2105	2106	2107	2110	2111	2112	2113
11	0	2114	2115	2116	2117	2120	2121	2122	2123	2124	2125
	1	2126	2127	2130	2131	2132	2133	2134	2135	2136	2137
	2	2140	2141	2142	2143	2144	2145	2146	2147	2150	2151
	3	2152	2153	2154	2155	2156	2157	2160	2161	2162	2163
	4	2164	2165	2166	2167	2170	2171	2172	2173	2174	2175
TENS	5	2176	2177	2200	2201	2202	2203	2204	2205	2206	2207
	6	2210	2211	2212	2213	2214	2215	2216	2217	2220	2221
	7	2222	2223	2224	2225	2226	2227	2230	2231	2232	2233
	8	2234	2235	2236	2237	2240	2241	2242	2243	2244	2245
	9	2246	2247	2250	2251	2252	2253	2254	2255	2256	2257

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

1200 Through 1399

UNITS

		0	1	2	3	4	5	6	7	8	9
12	0	2260	2261	2262	2263	2264	2265	2266	2267	2270	2271
	1	2272	2273	2274	2275	2276	2277	2300	2301	2302	2303
	2	2304	2305	2306	2307	2310	2311	2312	2313	2314	2315
	3	2316	2317	2320	2321	2322	2323	2324	2325	2326	2327
	4	2330	2331	2332	2333	2334	2335	2336	2337	2340	2341
TENS	5	2342	2343	2344	2345	2346	2347	2350	2351	2352	2353
	6	2354	2355	2356	2357	2360	2361	2362	2363	2364	2365
	7	2366	2367	2370	2371	2372	2373	2374	2375	2376	2377
	8	2400	2401	2402	2403	2404	2405	2406	2407	2410	2411
	9	2412	2413	2414	2415	2416	2417	2420	2421	2422	2423
13	0	2424	2425	2426	2427	2430	2431	2432	2433	2434	2435
	1	2436	2437	2440	2441	2442	2443	2444	2445	2446	2447
	2	2450	2451	2452	2453	2454	2455	2456	2457	2460	2461
	3	2462	2463	2464	2465	2466	2467	2470	2471	2472	2473
	4	2474	2475	2476	2477	2500	2501	2502	2503	2504	2505
TENS	5	2506	2507	2510	2511	2512	2513	2514	2515	2416	2517
	6	2520	2521	2522	2523	2524	2525	2526	2527	2530	2531
	7	2532	2533	2534	2535	2536	2637	2540	2541	2542	2543
	8	2544	2545	2546	2547	2550	2551	2552	2553	2554	2555
	9	2556	2557	2560	2561	2562	2563	2564	2565	2566	2567

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

1400 Through 1599

UNITS

		0	1	2	3	4	5	6	7	8	9
14	0	2570	2571	2572	2573	2574	2575	2576	2577	2600	2601
	1	2602	2603	2604	2605	2606	2607	2610	2611	2612	2613
	2	2614	2615	2616	2617	2620	2621	2622	2623	2624	2525
	3	2626	2627	2630	2631	2632	2633	2634	2635	2636	2637
	4	2640	2641	2642	2643	2644	2645	2646	2647	2650	2651
TENS	5	2652	2653	2654	2655	2656	2657	2660	2661	2662	2663
	6	2664	2665	2666	2667	2670	2671	2672	2673	2674	2675
	7	2676	2677	2700	2701	2702	2703	2704	2705	2706	2707
	8	2710	2711	2712	2713	2714	2715	2716	2717	2720	2721
	9	2722	2723	2724	2725	2726	2727	2730	2731	2732	2733
15	0	2734	2735	2736	2737	2740	2741	2742	2743	2744	2745
	1	2746	2747	2750	2751	2752	2753	2754	2755	2756	2757
	2	2760	2761	2762	2763	2764	2765	2766	2767	2770	2771
	3	2772	2773	2774	2775	2776	2777	3000	3001	3002	3003
	4	3004	3005	3006	3007	3010	3011	3012	3013	3014	3015
TENS	5	3016	3017	3020	3021	3022	3023	3024	3025	3026	3027
	6	3030	3031	3032	3033	3034	3035	3036	3037	3040	3041
	7	3042	3043	3044	3045	3046	3047	3050	3051	3052	3053
	8	3054	3055	3056	3057	3060	3061	3062	3063	3064	3065
	9	3066	3067	3070	3071	3072	3073	3074	3075	3076	3077

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

1600 Through 1799

		UNITS									
		0	1	2	3	4	5	6	7	8	9
16	0	3100	3101	3102	3103	3104	3105	3106	3107	3110	3111
	1	3112	3113	3114	3115	3116	3117	3120	3121	3122	3123
	2	3124	3125	3126	3127	3130	3131	3132	3133	3134	3135
	3	3136	3137	3140	3141	3142	3143	3144	3145	3146	3147
	4	3150	3151	3152	3153	3154	3155	3156	3157	3160	3161
TENS	5	3162	3163	3164	3165	3166	3167	3170	3171	3172	3173
	6	3174	3175	3176	3177	3200	3201	3202	3203	3204	3205
	7	3206	3207	3210	3211	3212	3213	3214	3215	3216	3217
	8	3220	3221	3222	3223	3224	3225	3226	3227	3230	3231
	9	3232	3233	3234	3235	3236	3237	3240	3241	3242	3243
17	0	3244	3245	3246	3247	3250	3251	3252	3253	3254	3255
	1	3256	3257	3260	3261	3262	3263	3264	3265	3266	3267
	2	3270	3271	3272	3273	3274	3275	3276	3277	3300	3301
	3	3302	3303	3304	3305	3306	3307	3310	3111	3312	3313
	4	3314	3315	3316	3317	3320	3321	3322	3323	3324	3325
TENS	5	3326	3327	3330	3331	3332	3333	3334	3335	3336	3337
	6	3340	3341	3342	3343	3344	3345	3346	3347	3350	3351
	7	3352	3353	3354	3355	3356	3357	3360	3361	3362	3363
	8	3364	3365	3366	3367	3370	3371	3372	3373	3374	3375
	9	3376	3377	3400	3401	3402	3403	3404	3405	3406	3407

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

1800 Through 1999

UNITS

		0	1	2	3	4	5	6	7	8	9
18	0	3410	3411	3412	3413	3414	3415	3416	3417	3420	342
	1	3422	3423	3424	3425	3426	3427	3430	3431	3432	3433
	2	3434	3435	3436	3437	3440	3441	3442	3443	3444	3445
	3	3446	3447	3450	3451	3452	3453	3454	3455	3456	3457
	4	3460	3461	3462	3463	3464	3465	3466	3467	3470	3471
TENS	5	3472	3473	3474	3475	3476	3477	3500	3501	3502	3503
	6	3504	3505	3506	3507	3510	3511	3512	3513	3514	3515
	7	3516	3517	3520	3521	3522	3523	3524	3525	3526	3527
	8	3530	3531	3532	3533	3534	3535	3536	3537	3540	3541
	9	3542	3543	3544	3545	3546	3547	3550	3551	3552	3553
19	0	3554	3555	3556	3557	3560	3561	3662	3563	3564	3565
	1	3566	3567	3570	3571	3572	3573	3574	3585	3576	3577
	2	3600	3601	3602	3603	3604	3605	3606	3607	3610	3611
	3	3612	3613	3614	3615	3616	3617	3620	3621	3622	3623
	4	3624	3625	3626	3627	3630	3631	3632	3633	3634	3635
TENS	5	3636	3637	3640	3641	3642	3643	3644	3645	3646	3647
	6	3650	3651	3652	3653	3654	3655	3656	3657	3660	3661
	7	3662	3663	3664	3665	3666	3667	3670	3671	3672	3673
	8	3674	3675	3676	3677	3700	3701	3702	3703	3704	3705
	9	3706	3707	3710	3711	3712	3713	3714	3715	3716	3717

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

2000 Through 2199

UNITS

		0	1	2	3	4	5	6	7	8	9
20	0	3720	3721	3722	3723	3724	3725	7326	3727	3730	3731
	1	3732	3733	3734	3735	3736	3737	3740	3741	3742	3743
	2	3744	3745	3746	3747	3750	3751	3752	3753	3754	3755
	3	3756	3757	3760	3761	3762	3763	3764	3765	3766	3767
	4	3770	3771	3772	3773	3774	3775	3776	3777	4000	4001
TENS	5	4002	4003	4004	4005	4006	4007	4010	4011	4012	4013
	6	4014	4015	4016	4017	4020	4021	4022	4023	4024	4025
	7	4026	4027	4030	4031	4032	4033	4034	4035	4036	4037
	8	4040	4041	4042	4043	4044	4045	4046	4047	4050	4051
	9	4052	4053	4054	4055	4056	4057	4060	4061	4062	4063
21	0	4064	4065	4066	4067	4070	4071	4072	4073	4074	4075
	1	4076	4077	4100	4101	4102	4103	4104	4105	4106	4107
	2	4110	4111	4112	4113	4114	4115	4116	4117	4120	4121
	3	4122	4123	4124	4125	4126	4127	4130	4131	4132	4133
	4	4134	4135	4136	4137	4140	4141	4142	4143	4144	4145
TENS	5	4146	4147	4150	4151	4152	4153	4154	4155	4156	4157
	6	4160	4161	4162	4163	4164	4165	4166	4167	4170	4171
	7	4172	4173	4174	4175	4176	4177	4200	4201	4202	4203
	8	4204	4205	4206	4207	4210	4211	4212	4213	4214	4215
	9	4216	4217	4220	4221	4222	4223	4224	4225	4226	4227

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

2200 Through 2399

UNITS

		0	1	2	3	4	5	6	7	8	9
22	0	4230	4231	4232	4233	4234	4235	4236	4237	4240	4241
	1	4242	4243	4244	4245	4246	4247	4250	4251	4252	4253
	2	4254	4255	4256	4257	4260	4261	4262	4263	4264	4265
	3	4266	4267	4270	4271	4272	4273	4274	4275	4276	4277
	4	4300	4301	4302	4303	4304	4305	4306	4307	4310	4311
TENS	5	4312	4313	4314	4315	4316	4317	4320	4321	4322	4323
	6	4324	4325	4326	4327	4330	4331	4332	4333	4334	4335
	7	4336	4337	4340	4341	4342	4343	4344	4345	4346	4347
	8	4350	4351	4352	4353	4354	4355	4356	4357	4360	4361
	9	4362	4363	4364	4365	4366	4367	4370	4371	4372	4373
23	0	4374	4375	4376	4377	4400	4401	4402	4403	4404	4405
	1	4406	4407	4410	4411	4412	4413	4414	4415	4416	4417
	2	4420	4421	4422	4423	4424	4425	4426	4427	4430	4431
	3	4432	4433	4434	4435	4436	4437	4440	4441	4442	4443
	4	4444	4445	4446	4447	4450	4451	4452	4453	4454	4455
TENS	5	4456	4457	4460	4461	4462	4463	4464	4465	4466	4467
	6	4470	4471	4472	4473	4474	4475	4476	4477	4500	4501
	7	4502	4503	4504	4505	4506	4507	4510	4511	4512	4513
	8	4514	4515	4516	4517	4520	4521	4522	4523	4524	4525
	9	4526	4527	4530	4531	4532	4533	4534	4535	4536	4537

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

2400 Through 2599

UNITS

		0	1	2	3	4	5	6	7	8	9
24	0	4540	4541	4542	4543	4544	4545	4546	4547	4550	4551
	1	4552	4553	4554	4555	4556	4557	4560	4561	4562	4563
	2	4564	4565	4566	4567	4570	4571	4572	4573	4574	4575
	3	4576	4577	4600	4601	4602	4603	4604	4605	4606	4607
	4	4610	4611	4612	4613	4614	4615	4616	4617	4620	4621
TENS	5	4622	4623	4624	4625	4626	4627	4630	4631	4632	4633
	6	4634	4635	4636	4637	4640	4641	4642	4643	4644	4645
	7	4646	4647	4650	4651	4652	4653	4654	4655	4656	4657
	8	4660	4661	4662	4663	4664	4665	4666	4667	4670	4671
	9	4672	4673	4674	4675	4676	4677	4700	4701	4702	4703
25	0	4704	4705	4706	4707	4710	4711	4712	4713	4714	4715
	1	4716	4717	4720	4721	4722	4723	4724	4725	4726	4727
	2	4730	4731	4732	4733	4734	4735	4736	4737	4740	4741
	3	4742	4743	4744	4745	4746	4747	4750	4751	4752	4753
	4	4754	4755	4756	4757	4760	4761	4762	4763	4764	4765
TENS	5	4766	4767	4770	4771	4772	4773	4774	4775	4776	4777
	6	5000	5001	5002	5003	5004	5005	5006	5007	5010	5011
	7	5012	5013	5014	5015	5016	5017	5020	5021	5022	5023
	8	5024	5025	5026	5027	5030	5031	5032	5033	5034	5035
	9	5036	5037	5040	5041	5042	5043	5044	5045	5046	5047

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

2600 Through 2799

		UNITS									
		0	1	2	3	4	5	6	7	8	9
26	0	5050	5051	5052	5053	5054	5055	5056	5057	5060	5061
	1	5062	5063	5064	5065	5066	5067	5070	5071	5072	5073
	2	5074	5075	5076	5077	5100	5101	5102	5103	5104	5105
	3	5106	5107	5110	5111	5112	5113	5114	5115	5116	5117
	4	5120	5121	5122	5123	5124	5125	5126	5127	5130	5131
TENS	5	5132	5133	5134	5135	5136	5137	5140	5141	5142	5143
	6	5144	5145	5146	5147	5150	5151	5152	5153	5154	5155
	7	5156	5157	5160	5161	5162	5163	5164	5165	5166	5167
	8	5170	5171	5172	5173	5174	5175	5176	5177	5200	5201
	9	5202	5203	5204	5205	5206	5207	5210	5211	5212	5213
27	0	5214	5215	5216	5217	5220	5221	5222	5223	5224	5225
	1	5226	5227	5230	5231	5232	5233	5234	5235	5236	5237
	2	5240	5241	5242	5243	5244	5245	5246	5247	5250	5251
	3	5252	5253	5254	5255	5256	5257	5260	5261	5262	5263
	4	5264	5265	5266	5267	5270	5271	5272	5273	5274	5275
TENS	5	5276	5277	5300	5301	5302	5303	5304	5305	5306	5307
	6	5310	5311	5312	5313	5314	5315	5316	5317	5320	5321
	7	5322	5323	5324	5325	5326	5327	5330	5331	5332	5333
	8	5334	5335	5336	5337	5340	5341	5342	5343	5344	5345
	9	5346	5347	5350	5351	5352	5353	5354	5355	5356	5357

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

2800 Through 2999

		UNITS									
		0	1	2	3	4	5	6	7	8	9
28	0	5360	5361	5362	5363	5364	5365	5366	5367	5370	5371
	1	5372	5373	5374	5375	5376	5377	5400	5401	5402	5403
	2	5404	5405	5406	5407	5410	5411	5412	5413	5414	5415
	3	5416	5417	5420	5421	5422	5423	5424	5425	5426	5427
	4	5430	5431	5432	5433	5434	5435	5436	5437	5440	5441
TENS	5	5442	5443	5444	5445	5446	5447	5450	5451	5452	5453
	6	5454	5455	5456	5457	5460	5461	5462	5463	5464	5465
	7	5466	5467	5470	5471	5472	5473	5474	5475	5476	5477
	8	5500	5501	5502	5503	5504	5505	5506	5507	5510	5511
	9	5512	5513	5514	5515	5516	5517	5520	5521	5522	5523
29	0	5524	5525	5526	5527	5530	5531	5532	5533	5534	5535
	1	5536	5537	5540	5541	5542	5543	5544	5545	5546	5547
	2	5550	5551	5552	5553	5554	5555	5556	5557	5560	5561
	3	5562	5563	5564	5565	5566	5567	5570	5571	5572	5573
	4	5574	5575	5576	5577	5600	5601	5602	5603	5604	5605
TENS	5	5606	5607	5610	5611	5612	5613	5614	5615	5616	5617
	6	5620	5621	5622	5623	5624	5625	5626	5627	5630	5631
	7	5632	5633	5634	5635	5636	5637	5640	5641	5642	5643
	8	5644	5645	5646	5647	5650	5651	5652	5653	5654	5655
	9	5656	5657	5660	5661	5662	5663	5664	5665	5666	5667

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

3000 Through 3199

		UNITS									
		0	1	2	3	4	5	6	7	8	9
30	0	5670	5671	5672	5673	5674	5675	5676	5677	5700	5701
	1	5702	5703	5704	5705	5706	5707	5710	5711	5712	5713
	2	5714	5715	5716	5717	5720	5721	5722	5723	5724	5725
	3	5726	5727	5730	5731	5732	5733	5734	5735	5736	5737
	4	5740	5741	5742	5743	5744	5745	5746	5747	5750	5751
TENS	5	5752	5753	5754	5755	5756	5757	5760	5761	5762	5763
	6	5764	5765	5766	5767	5770	5771	5772	5773	5774	5775
	7	5776	5777	6000	6001	6002	6003	6004	6005	6006	6007
	8	6010	6011	6012	6013	6014	6015	6016	6017	6020	6021
	9	6022	6023	6024	6025	6026	6027	6030	6031	6032	6033
31	0	6034	6035	6036	6037	6040	6041	6042	6043	6044	6045
	1	6046	6047	6050	6051	6052	6053	6054	6055	6056	6057
	2	6060	6061	6062	6063	6064	6065	6066	6067	6070	6071
	3	6072	6073	6074	6075	6076	6077	6100	6101	6102	6103
	4	6104	6105	6106	6107	6110	6111	6112	6113	6114	6115
TENS	5	6116	6117	6120	6121	6122	6123	6124	6125	6126	6127
	6	6130	6131	6132	6133	6134	6135	6136	6137	6140	6141
	7	6142	6143	6144	6145	6146	6147	6150	6151	6152	6153
	8	6154	6155	6156	6157	6160	6161	6162	6163	6164	6165
	9	6166	6167	6170	6171	6172	6173	6174	6175	6176	6177

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

3200 Through 3399

UNITS

		0	1	2	3	4	5	6	7	8	9
32	0	6200	6201	6202	6203	6204	6205	6206	6207	6210	6211
	1	6212	6213	6214	6215	6216	6217	6220	6221	6222	6223
	2	6224	6225	6226	6227	6230	6231	6232	6233	6234	6235
	3	6236	6237	6240	6241	6242	6243	6244	6245	6246	6247
	4	6250	6251	6252	6253	6254	6255	6256	6257	6260	6261
TENS	5	6262	6263	6264	6265	6266	6267	6270	6271	6272	6273
	6	6274	6275	6276	6277	6300	6301	6302	6303	6304	6305
	7	6306	6307	6310	6311	6312	6313	6314	6315	6316	6317
	8	6320	6321	6322	6323	6324	6325	6326	6327	6330	6331
	9	6332	6333	6334	6335	6336	6337	6340	6341	6342	6343
33	0	6344	6345	6346	6347	6350	6351	6352	6353	6354	6355
	1	6356	6357	6360	6361	6362	6363	6364	6365	6366	6367
	2	6370	6371	6372	6373	6374	6375	6376	6377	6400	6401
	3	6402	6403	6404	6405	6406	6407	6410	6411	6412	6413
	4	6414	6415	6416	6417	6420	6421	6422	6423	6424	6425
TENS	5	6426	6427	6430	6431	6432	6433	6434	6435	6436	6437
	6	6440	6441	6442	6443	6444	6445	6446	6447	6450	6451
	7	6452	6453	6454	6455	6456	6457	6460	6461	6462	6463
	8	6464	6465	6466	6467	6470	6471	6472	6473	6474	6475
	9	6476	6477	6500	6501	6502	6503	6504	6505	6506	6507

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

3400 Through 3599

						UNITS					
		0	1	2	3	4	5	6	7	8	9
34	0	6510	6511	6512	6513	6514	6515	6516	6517	6520	6521
	1	6522	6523	6524	6525	6526	6527	6530	6531	6532	6533
	2	6534	6535	6536	6537	6540	6541	6542	6543	6544	6545
	3	6546	6547	6550	6551	6552	6553	6554	6555	6556	6557
	4	6560	6561	6562	6563	6564	6565	6566	6567	6570	6571
TENS	5	6572	6573	6574	6575	6576	6577	6600	6601	6602	6603
	6	6604	6605	6606	6607	6610	6611	6612	6613	6614	6615
	7	6616	6617	6620	6621	6622	6623	6624	6625	6626	6627
	8	6630	6631	6632	6633	6634	6635	6636	6637	6640	6641
	9	6642	6643	6644	6645	6646	6647	6650	6651	6652	6653
35	0	6654	6655	6656	6657	6660	6661	6662	6663	6664	6665
	1	6666	6667	6670	6671	6672	6673	6674	6675	6676	6677
	2	6700	6701	6702	6703	6704	6705	6706	6707	6710	6711
	3	6712	6713	6714	6715	6716	6717	6720	6721	6722	6723
	4	6724	6725	6726	6727	6730	6731	6732	6733	6734	6735
TENS	5	6736	6737	6740	6741	6742	6743	6744	6745	6746	6747
	6	6750	6751	6752	6753	6754	6755	6756	6757	6760	6761
	7	6762	6763	6764	6765	6766	6767	6770	6771	6772	6773
	8	6774	6775	6776	6777	7000	7001	7002	7003	7004	7005
	9	7006	7007	7010	7011	7012	7013	7014	7015	7016	7017

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

3600 Through 3799

		UNITS									
		0	1	2	3	4	5	6	7	8	9
36	0	7020	7021	7022	7023	7024	7025	7026	7027	7030	7031
	1	7032	7033	7034	7035	7036	7037	7040	7041	7042	7043
	2	7044	7045	7046	7047	7050	7051	7052	7053	7054	7055
	3	7056	7057	7060	7061	7062	7063	7064	7065	7066	7067
	4	7070	7071	7072	7073	7074	7075	7076	7077	7100	7101
TENS	5	7102	7103	7104	7105	7106	7107	7110	7111	7112	7113
	6	7114	7115	7116	7117	7120	7121	7122	7123	7124	7125
	7	7126	7127	7130	7131	7132	7133	7134	7135	7136	7137
	8	7140	7141	7142	7143	7144	7145	7146	7147	7150	7151
	9	7152	7153	7154	7155	7156	7157	7160	7161	7162	7163
37	0	7164	7165	7166	7167	7170	7171	7172	7173	7174	7175
	1	7176	7177	7200	7201	7202	7203	7204	7205	7206	7207
	2	7210	7211	7212	7213	7214	7215	7216	7217	7220	7221
	3	7222	7223	7224	7225	7226	7227	7230	7231	7232	7233
	4	7234	7235	7236	7237	7240	7241	7242	7243	7244	7245
TENS	5	7246	7247	7250	7251	7252	7253	7254	7255	7256	7257
	6	7260	7261	7262	7263	7264	7265	7266	7267	7270	7271
	7	7272	7273	7274	7275	7276	7277	7300	7301	7302	7303
	8	7304	7305	7306	7307	7310	7311	7312	7313	7314	7315
	9	7316	7317	7320	7321	7322	7323	7324	7325	7326	7327

DECIMAL TO OCTAL CONVERSION TABLE (cont.)
(0-4096)

3800 Through 3999

		UNITS									
		0	1	2	3	4	5	6	7	8	9
38	0	7330	7331	7332	7333	7334	7335	7336	7337	7340	7341
	1	7342	7343	7344	7345	7346	7347	7350	7351	7352	7353
	2	7354	7355	7356	7357	7360	7361	7362	7363	7364	7365
	3	7366	7367	7370	7371	7372	7373	7374	7375	7376	7377
	4	7400	7401	7402	7403	7404	7405	7406	7407	7410	7411
TENS	5	7412	7413	7414	7415	7416	7417	7420	7421	7422	7423
	6	7424	7425	7426	7427	7430	7431	7432	7433	7434	7435
	7	7436	7437	7440	7441	7442	7443	7444	7445	7446	7447
	8	7450	7451	7452	7453	7454	7455	7456	7457	7460	7461
	9	7462	7463	7464	7465	7466	7467	7470	7471	7472	7473
39	0	7474	7475	7476	7477	7500	7501	7502	7503	7504	7505
	1	7506	7507	7510	7511	7512	7513	7514	7515	7516	7517
	2	7520	7521	7522	7523	7524	7525	7526	7527	7530	7531
	3	7532	7533	7534	7535	7536	7537	7540	7541	7542	7543
	4	7544	7545	7546	7547	7550	7551	7552	7553	7554	7555
TENS	5	7556	7557	7560	7561	7562	7563	7564	7565	7566	7567
	6	7570	7571	7572	7573	7574	7575	7576	7577	7600	7601
	7	7602	7603	7604	7605	7606	7607	7610	7611	7612	7613
	8	7614	7615	7616	7617	7620	7621	7622	7623	7624	7625
	9	7626	7627	7630	7631	7632	7633	7634	7635	7636	7637

DECIMAL TO OCTAL CONVERSION TABLE (concl.)
(0-4096)

4000 Through 4096

		UNITS									
		0	1	2	3	4	5	6	7	8	9
40	0	7640	7641	7642	7643	7644	7645	7646	7647	7650	7651
	1	7652	7653	7654	7655	7656	7657	7660	7661	7662	7663
	2	7664	7665	7666	7667	7670	7671	7672	7673	7674	7675
	3	7676	7677	7700	7701	7702	7703	7704	7705	7706	7707
	4	7710	7711	7712	7713	7714	7715	7716	7717	7720	7721
TENS	5	7722	7723	7724	7725	7726	7727	7730	7731	7732	7733
	6	7734	7735	7736	7737	7740	7741	7742	7743	7744	7745
	7	7746	7747	7750	7751	7752	7753	7754	7755	7756	7757
	8	7760	7761	7762	7763	7764	7765	7766	7767	7770	7771
	9	7772	7773	7774	7775	7776	7777	10000			