manual of operations

THE CENTRAL COMPUTER OF THE UNIVAC SYSTEM

prepared by the Training Section,
Electronic Computer Department
Remington Rand

EL · 210

MANUAL OF OPERATION
FOR
UNIVAC SYSTEM

MANUAL OF OPERATION
FOR UNIVAC SYSTEM

Table of Contents

PREFACE

The purpose of this manual is to serve as a reference text for personnel learning operational procedures of the UNIVAC System. It was not intended as an encyclopedic reference to the System, and in actuality provides operational information for the central computer only. No prior knowledge of UNIVAC or other large scale digital computers is presumed.

The first four chapters provide a general introduction to the central computer and may be of interest to readers other than operators. The elements of programming, logical design, and the order code are described. Since the logical design of the UNIVAC is exceedingly complex it would be quite outside the scope of this manual. A knowledge of logic is essential to meaningful understanding of the operation of the computer however, and is the reason for describing a simple computer in Chapter II. The elements of UNIVAC are then introduced by analogy with the components of the simpler computer.

The final three chapters comprise a description of the Supervisory Control Console and normal and error diagnostic procedures. Only the more common procedures in error diagnosis are described, and the point of view expressed is that all cases which are ambivalent in interpretation should be handled by instituting a general rerun. Descriptions of the Service Routines, which are of great convenience to the operation of a UNIVAC, are not included as they are already described in another manual and are continually being modified by further operating experience.

CHAPTER I

DESCRIPTION OF UNIVAC AND
INTRODUCTION TO PROGRAMMING

SECTION I

Concepts of a Computer

The UNIVAC is a large electronic device designed to perform repetitive clerical or mathematical operations at a high rate of speed and accuracy. In order to make clear the function of each component of the computer and its relationship to other components, let us consider the operation of a payroll clerk.

The clerk has a stack of time cards showing the number of hours a company's employees have worked during the week. She has a second stack of cards which show for each employee certain bits of fundamental information such as the hourly rate of pay, dependents, etc. As a work aid she uses a calculating machine which performs addition, subtraction, and multiplication for her.

In addition, she herself provides a factor of control or procedure, she selects the time card and hourly rate card for a particular employee. By means of the calculator she computes the man's gross pay, subtracts his income tax, old age insurance tax, etc., and ends up with the man's net pay which she enters in her ledger book.

This payroll process can be thought of in terms of four basic operations: Input, Output, Arithmetic, and Supervision. As shown in figure 1, the time cards and hourly rate cards are the input data, the calculator is the arithmetic unit, and the output is the net pay entered in the ledger. The whole operation is supervised and executed by the clerk.

If we think a bit more about the position of the clerk in this process, we recognize that her ability to turn the input of time cards and hourly rate cards into the ledger entries of net pay depends upon her remembering the individual steps of the operation -- the rules; that is, she must remember that she is to subtract the income tax from the gross pay, and that the income tax is computed by multiplying the number of dependents by $600 and so forth. Her supervision then actually consists of <u>memorizing</u> the necessary steps and <u>controlling</u> the order of their execution. This memory function becomes more apparent if the clerk has other duties to perform as well. For instance, she may be expected to maintain the hourly rate cards, adding or removing dependents upon notice, or changing the hourly rate when a man is given a raise.

When we try to mechanize this process, we have the option of building the supervision into the device so that it will do only payroll calculations, or of building into it the ability to do a small number of fundamental operations and then store in a memory the proper sequence of performing these operations. A computer with built-in supervision or program is called a Special Purpose Computer, while a computer which stores its supervision in a memory and which allows that supervision to be easily altered is termed a General Purpose Computer.

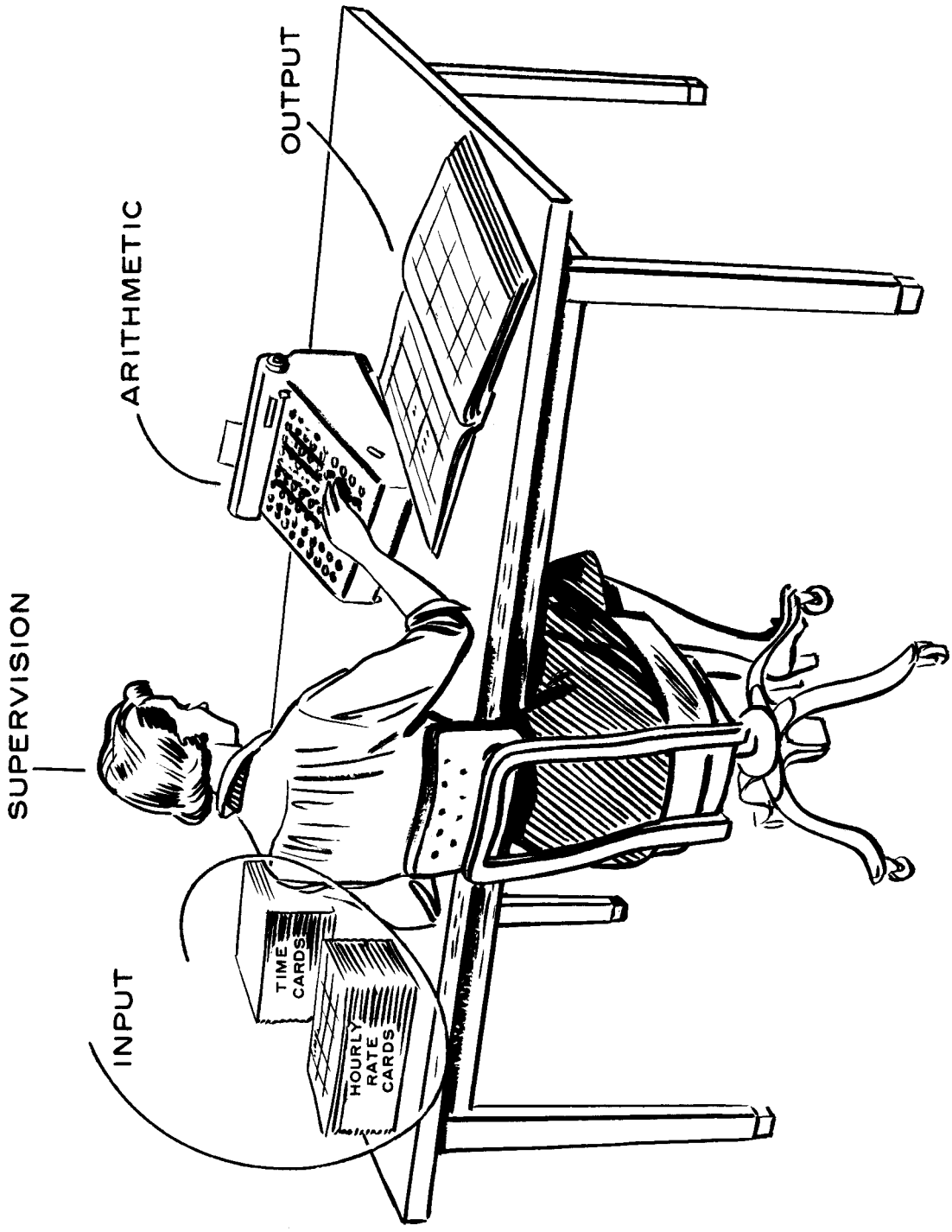

OUTPUT
ARITHMETIC
SUPERVISION
INPUT
TIME CARDS
HOURLY RATE CARDS

FIGURE 1

The UNIVAC is such a General Purpose Computer. Its five components and their interrelations are shown in figure 2. The memory occupies a central position. It holds for processing the information which has been read by the input device. It also holds the instructions that tell what to do with this input information. The control unit selects each instruction from the memory and executes it. An instruction might call for the hourly rate and total hours worked to be sent to the arithmetic unit for multiplication, and return the product to the memory. As the net wages are computed, they are sent to the output device for permanent recording.

In order to secure high operating speeds these five components are of electronic construction, and to assure complete accuracy many of the units are duplicated and cross checked. There are other checking methods used when it is not feasible to duplicate equipment.

There is yet another method of categorizing computers which is intimately related to the concept of memory. The basic element being processed by a computer is a number. We are all familiar with two methods of representing numbers.

1) The digital method, probably the oldest, uses a special symbol or mark to represent each number. For example, we use the Arabic symbols with the position concept when we indicate that a dozen is "12" and a dozen dozen, gross, is "144".

2) The analog method is familiar to us in the slide rule, where numbers are represented by different distances along a stick. An ammeter represents the number of amperes of current flow by the angle of a pointer. Other commonly used representations are electric current flow, rotation of a gear or shaft, and extension of a spring.

Computers may then be of two general types, special purpose or general purpose, depending on whether or not they have a stored program of instructions. Within each type a computer may be digital or analog depending upon the manner of representing numbers.

The UNIVAC is a General Purpose Digital Computer.

SECTION 2

Organization of UNIVAC Memory

To understand the operation of the UNIVAC we start with the central element of the five, the memory. Delaying until later a description of the physical make-up of the memory, we will describe its general characteristics as concern the user of the computer. The basic unit of memory in the UNIVAC is the "word". A word always consists of twelve characters or digits. There are 63 characters available; any combination of twelve of these constitutes a word. The 63 characters are listed in the appendix for reference. For our immediate needs, however, the numbers 0,1,2,...,9 and the letters of the alphabet A,B.C,...Z, and the minus sign (-) are among the 63. Here are some typical "words":

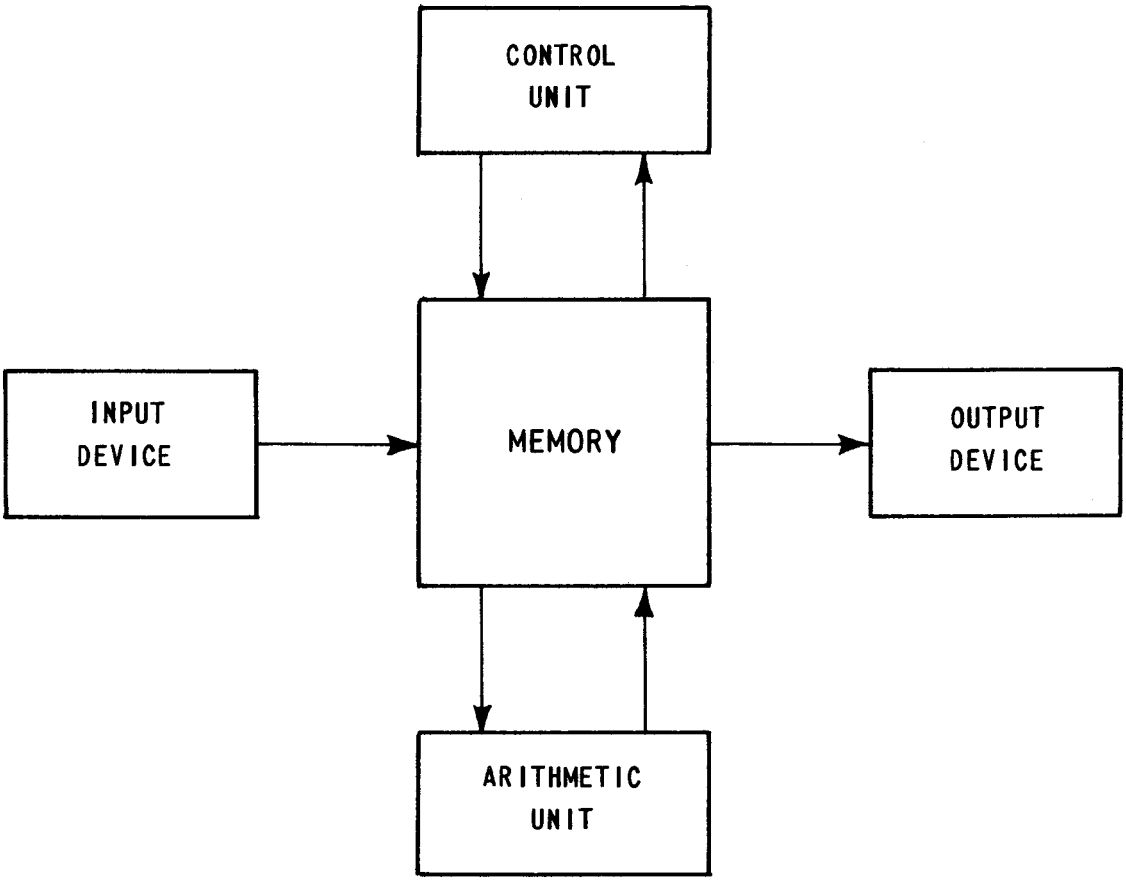

CONTROL
UNIT
INPUT
DEVICE
MEMORY
OUTPUT
DEVICE
ARITHMETIC
UNIT

FIGURE 2

012345678906

-00125000000

ABCDEFG986KL

JOHNNY-JONES

The main memory of UNIVAC has the capacity to "store" or "remember" 1000 such words. The memory may be visualized by thinking of 1000 boxes and in each box is a slip of paper with a UNIVAC word written on it (figure 3). In order to locate a particular word, we number or "address" the boxes. In UNIVAC the boxes are numbered consecutively from 000 to 999. Thus we can speak of the word in box 001 as being Johnny-Jones.

Earlier we mentioned that a General Purpose Computer stores the procedure or program it is to follow in the memory. The program consists of a number of elementary operations which are combined in such a way as to produce the desired answer. These elementary operations are called instructions. There are 43 different instructions to which the UNIVAC will respond. Each instruction requires six characters for its specification, for example:

B00 934

The six characters are designated from left to right as "first instruction digit", "second instruction digit", "third instruction digit", ... "sixth instruction digit". Generally, the fourth, fifth, and sixth instruction digits are the address of a particular box or memory cell, and the first and second instruction digits tell what operation is to be performed on the word in that cell. The third instruction digit plays no part in the operation of the instruction. Since each instruction consists of six characters, a UNIVAC word can consist of two instructions. The six characters on the left-hand side of a word are called the left instruction, and the right-hand six are called the right instruction.

In addition to the 1000-word main memory there are some special memory registers. These registers may be divided into three groups: 1) Arithmetic Registers, 2) Control Registers and 3) Transfer Registers.

Arithmetic Registers	Register A, rA, which holds one word Register x, rX, " " " " Register L, rL, " " " " Register F, rF, " " " "
Control Registers	Control Counter, CC, a one-word register Control Register, CR," " " " Static Register, SR, " half-word register
Transfer Registers	Register V, rV, a two-word register Register Y, rY, a ten-word register Register I, rI, a 60-word register Register O, rO, " " " "

SIMPLIFIED VISUAL CONCEPT

OF UNIVAC MEMORY

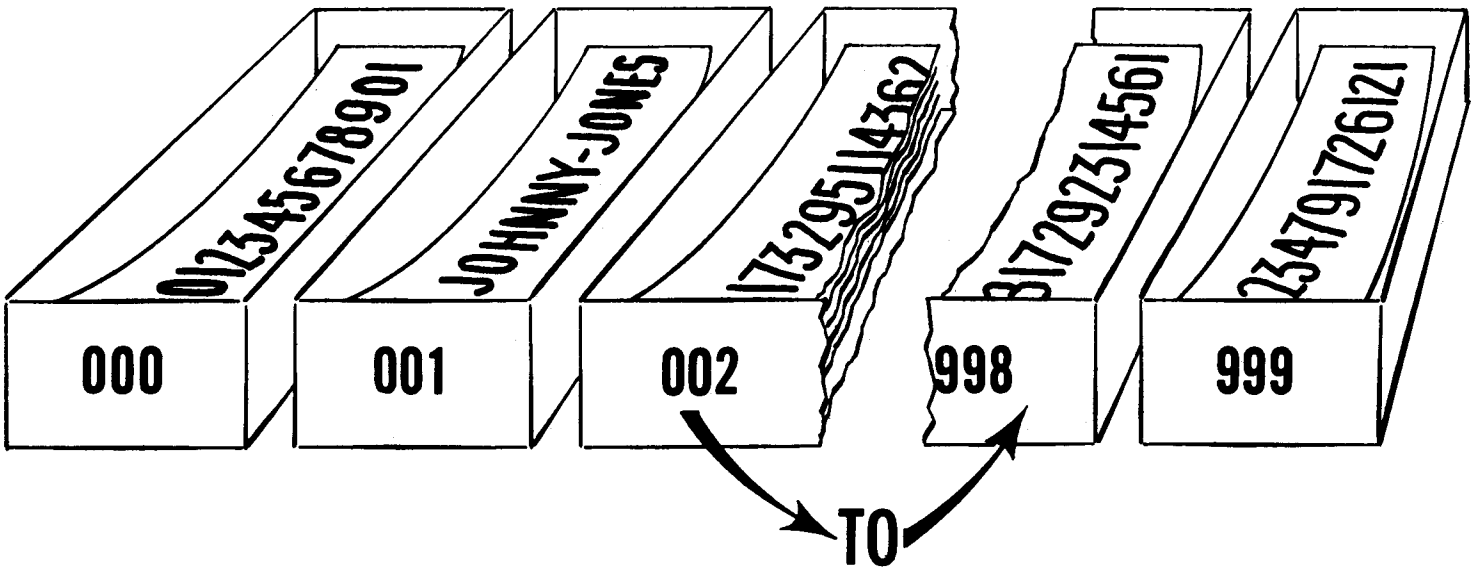


FIGURE 3

SECTION 3

Mode of Operation

The basic principle of operation of a General Purpose Computer, that is, a computer whose operating instructions or program are stored in the memory, consists of two general parts: a) Locate the instruction to be executed, and b) Execute it.

In UNIVAC, the Control Counter, a one-word register, contains the address (i.e. location) of the next instruction word. The three address digits are on the extreme right: 000000000XXX, the X's stand for any number between 000 and 999. Since each word may hold two instructions, each look-up in the memory extracts two instructions.

The basic cycle of operation in UNIVAC consists of four stages. Each stage is assigned a letter of the Greek alphabet to identify it. The four-stage cycle is executed in the order:

Stage	Description
α (Stage 1)	Send the right-hand six digits of the word in the Control Counter to the Static Register.
β (Stage 2)	Extract the word from the memory location specified by the right-hand three digits in the Static Register and place it in the Control Register, CR. Send the word in the Control Counter to the adder, add one to it, and return the sum to CC.
γ (Stage 3)	Send the left-hand six digits of the word in the Control Register to the Static Register and, treating it as an instruction, execute it.
δ (Stage 4)	Send the right-hand six digits of the word in CR to SR and execute it as an instruction.

This cycle then repeats itself.

As can be seen, if the Control Counter initially reads 000000000000 the instructions will be executed sequentially, first the left-hand instruction in cell 000, then the right-hand one. The next instruction is the left-hand one in cell 001, then the right-hand one in that cell, then the left-hand one in 002, etc.

As will be seen, there are ways to break this sequence through the instruction itself. Also, it is apparent from the description of the four-stage cycle that the instructions are executed in the Static Register only. Since the Static

Register can hold only half a word, the Control Register is required to store the right-hand instruction while the left one is being executed. The operating speed of a computer is largely governed by the amount of memory look-up; thus, by having a Control Register we need only one look-up to select two instructions.

SECTION 4

Elementary UNIVAC Instructions

At this point it will be helpful to introduce some basic instruction (orders) and prepare a few sample problems. This will tie together the foregoing concepts. Generally speaking each of the 43 UNIVAC instructions can be thought of as belonging to one of three groups:

A) Instructions which transfer information from one memory location to another or from one media to another.

B) Instructions which perform the arithmetic operations of addition, subtraction, multiplication and division.

C) Instructions which allow a choice between following one of two sequences of instructions.

Most of the UNIVAC instructions are redundant. That is, many of the instructions can be done by combinations of other instructions. The particular instructions for the UNIVAC were selected after studying many applications to determine which would most enhance the operating speed. Under Group A we will make use of the three instructions B m, C m, and L m.

B m instruction: This instruction when it is in the Static Register* causes the UNIVAC to extract the word in memory cell m and place it in rA and rX, wiping out the previous contents of those two registers. The word will still be present in memory cell m. This last statement is true for all UNIVAC orders which transfer information out of the main memory.

As an aid in memorizing the code it will be noted that the orders are of mnemonic character. Thus "B m" can be read as "Bring the contents of memory location m into rA and rX."

The complete six digit appearance of the B m instruction is: B00XXX where the X's refer to some memory location between 000 and 999. Some, but not all, instructions require specification of the second instruction digit. Those that do not usually have a zero in this digit position. No instruction requires specification of the third instruction digit and this position usually is written as zero also. To save effort in writing the instructions, these digits are usually omitted when they needn't be specified.

L m instruction: This is very similar to B m. It causes UNIVAC to transfer the word in memory location m to rL and rX, erasing the previous contents of those two registers.

* This statement holds for all instructions. They must be in the Static Register during γ or δ stages in order to be executed as an instruction.

C m instruction: In executing this instruction the UNIVAC erases the contents of memory location m and places in this location the contents of rA. After the transfer, the contents of rA are replaced with decimal zeros.

The only instruction in group B we shall consider at this time is the A m.

A m instruction: This is executed in two steps:

1) The word in memory location m is transferred to rX, the previous contents of rX being erased.

2) Then the contents of rA, and rX are sent to the algebraic adder, added, and the sum returned to rA.

The addition in step 2 is algebraic. The signs of the two numbers being in the left-most digit positions. For example, the proper sum of 012345 678905* and -01 111 111 111 is 001 234 567 794. Also any alphabetic character is passed on to the sum unchanged:

012 456 014 892	A00 006 C00 125
032 A69 CZ1 54G	007 B03 198 00Z
044 A25 CZ6 43G	A07 B09 C98 12Z

Among the group C instructions are the

00 m instruction: This requires a zero as the second instruction digit. This is the skip order. It tells UNIVAC to pass directly to the next instruction without altering any registers or memory cells. Address m is ignored.

9 m instruction: This is the stop order. It causes UNIVAC to stop executing instructions, leaving the registers and memory unaltered.

U m instruction: The orders previously discussed could be performed as either right- or left-hand instructions. This is true for all UNIVAC instructions except the U m, Q m, and T m orders. These should be right-hand instructions to perform properly. U m is an unconditional transfer of control instruction. It tells the UNIVAC to erase the three right-hand digits of the word in the Control Counter (the address of the next instruction pair) and put in their place the three digits of the memory location part of the U order. That is, if the order is U00 325 and the Control Counter contains 000 000 000 114, the "114" is erased and replaced by the "325". This and the Q order below are two means of breaking the sequential execution of orders.

Q m instruction: This is a conditional transfer of control. If the contents of rA and rL are identical the Q instruction is interpreted as a U order. That is, if (rA)=(rL)** the memory address digits m replace the address digits in the Control Counter. If (rA)≠(rL) the Q m is interpreted as a skip.

* A zero in the sign position is considered a plus sign.

** (rA) means the contents of, or the word in Register A
(015) " " " " " " " " memory cell 015

SECTION 5

Simple Problems

As our beginning problem in simple coding, let us assume that memory location 100 contains a quantity A and memory location 101 contains a quantity B. Let us design a program that permits the computer to interchange A and B and then stops the computer. We shall start our program in memory location 000 for convenience. Let us also not concern ourselves as to how the program is placed in the memory, but assume instead that everything we need will be placed in the cells indicated.

Memory Cell	Left Instruction	Right Instruction
000	B00 100	C00 102
001	B00 101	C00 100
002	B00 102	C00 101
003	900 000	000 000

This program is elementary, yet it shows how the instructions are strung together to obtain the desired results. Here a temporary storage place for the contents of cell 100 (A) are needed while we place B, the word in 101, in cell 100. Then we can take A out of temporary storage and place it in cell 101. All we need do to execute the program is to start the computer on α time with 000 000 000 000 in the Control Counter.

A more advanced program is the following: memory cells 100 - 999 contain a set of numbers. We wish to add these 900 numbers together, place the sum in memory location 099 and stop the computer. Here the numbers to be added are deliberately chosen to be too large to be written out in detail, that is, to code 899 additions. A method of doing the problem is illustrated in the following program: (The left-hand instruction and right-hand instruction are staggered to allow space for explanatory remarks.)

Memory Cell	L H I	R H I	Remarks
000	C00 099		
		C00 099	Zeros 099
001	B00 099		
		A00 100	Add first number to 099
002	C00 099		
		B00 001	
003	L00 007		Go to cell 006 if the number in
		Q00 006	cell 999 has been added
004	A00 008		
		C00 001	Increase the instruction in cell 001 to add the next number
005	000 000		
		U00 001	
006	900 000		Stop the computer
		000 000	
007	B00 099		
		A00 999	
008	000 000		
		000 001	

This example illustrates an important concept of programming, the iterative routine. Initially we place zeros in cell 099 and then put down the instructions necessary to add the first of our 900 numbers. Next we alter these instructions so that they will be able to add the second number to the partial sum. Then by repeating the addition process we will have added the second number, then the third, and so forth. In order to stop the computer we must know when the last number has been added. We will have added this last number when cell 001 contains the instruction B00 099 A00 999.

By examining cell 001 after adding a number to the partial sum we can stop when the last number has been added.

This problem illustrates one of the most useful characteristics of a stored program computer - the ability to subject its instructions to the same arithmetic operations as it does its data, that is, the power to alter its instructions.

Notice that the contents of cell 007 looks like a pair of instructions. But they are never executed as instructions. Even though they are used in the program they never get in the Static Register and so can never be treated as computer orders. In this type of computer, then, the distinction between instructions and data is based solely upon which one gets into the Static Register. This distinction is not made by the position of the instructions and data in the memory nor by their appearance.

CHAPTER 11

INTRODUCTION TO COMPUTER OPERATIONS

SECTION 1

Preliminary Discussion

The purpose of this Chapter is to present information which will lead to an understanding of the fundamental operation of UNIVAC. It should be clear that the scope of this development will be on an elementary level. In pursuance of this idea it is planned to discuss in some detail a simple computer which uses components contained in UNIVAC, but, it must be emphasized, this computer will not be UNIVAC. The following chapter will point out the similarities and differences that exist between UNIVAC and the computer herein developed. However, before describing the computer and its relationship to the operation of UNIVAC, it is advisable to review some elementry concepts of electricity.

SECTION 2

Basic Electrical Concepts

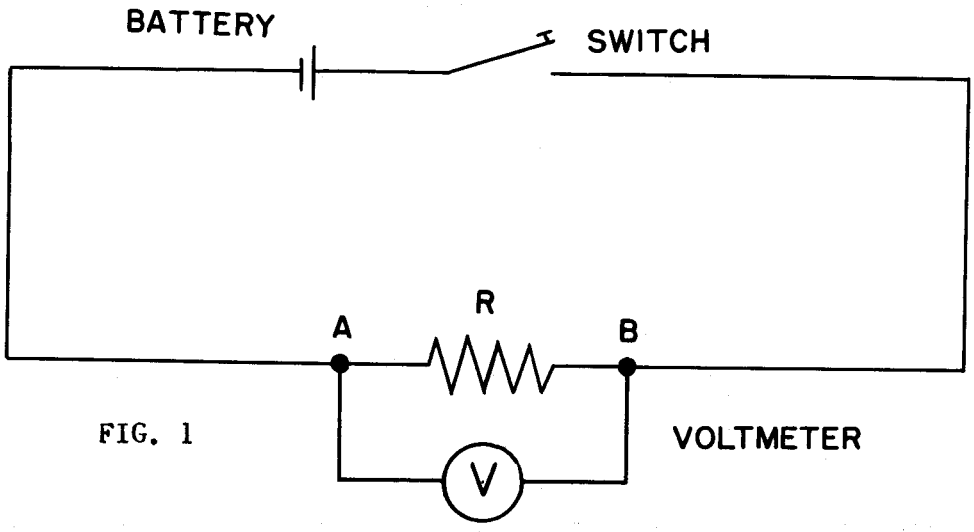


FIG. 1

The simplest form of electrical circuit is a battery with a resistance connected to its terminals as shown in Fig. 1. A complete circuit must have an unbroken path so that current can flow out of the battery, through the elements connected to it and back into the battery. The circuit is broken or opened if some part of this path is removed. A switch is a device for opening or closing the circuit and, hence, for preventing or allowing current to flow.

The battery in this circuit maintains a difference of potential energy, V (measured in volts), between its terminals which will force current to flow through the circuit elements. The current, I, is defined as the movement of electricity along a conductor and is measured in amperes. It has been found that the current flowing in a circuit is proportional to the difference of potential established between the terminals of the battery. Then $V \propto I$.

This can be written as an equation by introducing a constant of proportionality R. The constant R (measured in ohms) is called the resistance of the circuit and measures the resistance to the flow of current through any given circuit. The expression for difference of potential between two points can be written as $V_{(volts)} = I_{(amps)} \times R_{(ohms)}$ and this relationship is called Ohm's Law.

Thus in Fig. 1, the voltage difference between points A and B can be determined by Ohm's Law if the current in amperes flowing between A and B and the resistance in ohms of this part of the circuit are known, or the voltage difference between points A and B can be measured directly by connecting a device called a voltmeter as shown in Fig. 1.

Suppose that, when the switch is closed, the battery forces a current flow of 2 amperes and that the resistance R is known to be 5 ohms. Then by Ohm's Law it will be known that the voltmeter will register 2 amp x 5 ohm = 10 volts. However, if the switch is opened the circuit is broken and no current flows. The difference of voltage between points A and B would now be 0 amp x 5 ohm = 0 volts.

Thus, by opening and closing the switch, the voltmeter can be made to register either zero or ten volts. Because the battery produces a constant potential difference and the resistance, R, does not vary the voltmeter can assume only one of these two values depending upon the position of the switch.

It is possible to hold point A at a constant potential level, and use this as a reference level. If this is chosen as zero, reference can be made directly to the voltage at point B. For, if the switch is closed, the difference of potential between A and B is ten volts. But since A is fixed at zero then point B must be at ten volts. Considering the circuit in Fig. 1, the voltage of point B will fluctuate between zero and ten volts as the switch is opened or closed.

SECTION 3

Representation of Information

The two possible voltage levels of point B provide a means of representing information. For example, the ten volt condition might represent true, the zero condition false, or the choice yes or no, or the number one or zero. It would seem, however, that the type and quantity of information that can be represented in this manner is limited. This is certainly true if point B is examined at only one instant of time. However, if point B is sampled one second after some reference time and then two seconds after the reference time, making the restriction that the person operating the switch cannot change its position more than once per second, four conditions can be represented.

Referring to Fig. 2, showing a graph of the voltage of point B plotted against time in seconds, we see that point B can have the value 0 volts during the first second, 0 volts during the second second, or ten volts during the first and zero during the second and so on. Four distinct patterns can be used to represent the decimal numbers 0,1,2,3.

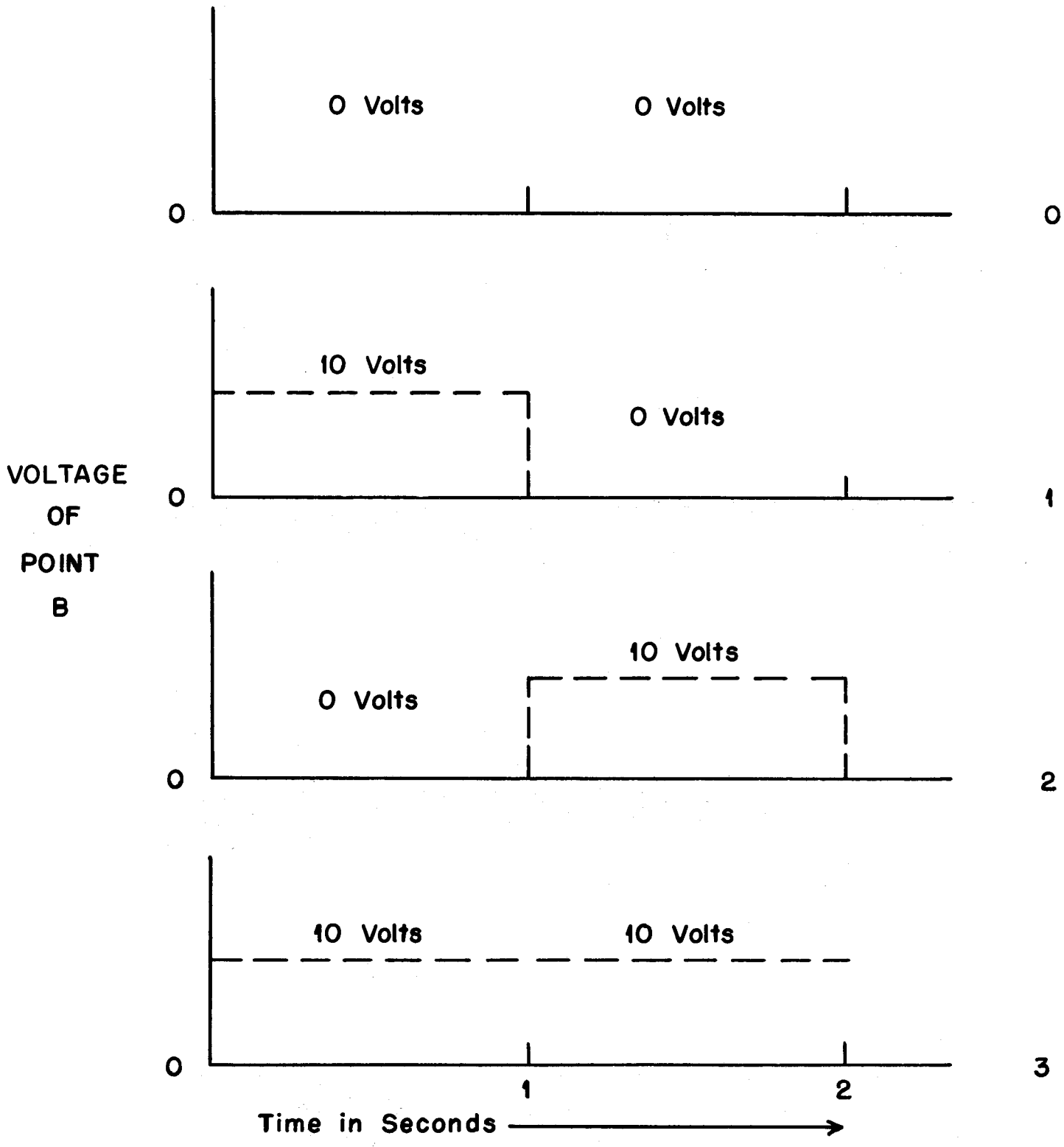

0 Volts
0 Volts
0
0
10 Volts
0 Volts
0
1
VOLTAGE
OF
POINT
B
0 Volts
10 Volts
0
2
10 Volts
10 Volts
0
3
1
2
Time in Seconds

FIG. 2

Sampling each of three successive time periods would produce one of 2^3 or eight possible patterns representing the decimal numbers zero through seven. By sampling n successive time periods one of 2^n possible patterns would exist. Thus, by choosing the number of time periods sufficiently large, any decimal number can be represented.

It is important to realize that the pattern of voltages of point B representing a number does not appear instantaneously but appears serially. This is necessary because point B can have only one of two possible voltages at any instant of time.

It is advantageous at this point to review the structure of the decimal number system. It is called the decimal system because it uses the ten integers 0, 1, 2 . . . 9. Furthermore, a number such as 239 is really a short hand method of writing 2 hundreds plus 3 tens plus 9 units. This may also be written as $2 \times (10^2) + 3 \times (10^1) + 9 \times (10^0)$, recalling that $10^0 = 1$, $10^1 = 10$, $10^2 = 10 \times 10 = 100$, $10^3 = 10 \times 10 \times 10 = 1000$, etc. The digits 2, 3, and 9 are really the coefficients of three different powers of <u>ten</u> which is called the <u>base</u> of the system. Again 2,134 is really a short way of writing $2 \times (10^3) + 1 \times (10^2) + 3 \times (10^1) + 4 \times (10^0)$.

There is no reason why ten must be the only base that can be used. Examine briefly the binary number system which uses only <u>two</u> integers 0 and 1. Numbers can be expressed in the same general form as in the decimal system. Thus 1101 in the binary system is really a short hand notation for writing.

$$1 \times (2^3) + 1 \times (2^2) + 0 \times (2^1) + 1 \times (2^0).$$

Its decimal equivalent would be.

$$8 + 4 + 0 + 1 = 13.$$

To represent the number expressed decimally as 17 would require one sixteen and one unit and would be written in binary as

$$1 \times (2^4) + 0 \times (2^3) + 0 \times (2^2) + 0 \times (2^1) + 1 \times (2^0) \text{ or } 10001.$$

It is interesting to note the similarity in the short hand notation for the binary representation of numbers and the voltage patterns representing numbers in the circuit in Fig. 2. In the binary system numbers are represented by a <u>serial</u> pattern of zeros and ones, and in the circuits by a <u>serial</u> pattern of no voltage and voltage. Thus, there is a direct correspondence between an easily produced characteristic of an electrical circuit and the binary system of representing numbers. This and the fact that the rules of binary arithmetic are very simple make it practicable to use simple voltage patterns to represent information for computational purposes.

Using the circuit described and the time interval of one second to represent a binary digit, it would take ten seconds to represent ten binary digits. Since such a pattern can represent only the range of numbers expressed decimally as 0 . . . . to 1023, it is evident that such a serial representation of information is quite time consuming.

The obvious remedy would be to decrease the time required to represent a binary digit and hence reduce the overall time to represent a given amount of information. If the time interval mentioned above is reduced from one second to 1/100 of a second, any ten binary digit number can be represented in 1/10 second instead of ten seconds.

It is impossible, however, to manually operate a switch this rapidly or to observe voltmeter readings changing so frequently. It is possible to replace the switch with an electrical device which will open or close the circuit rapidly for accurately measured time intervals. With such a rapidly operating device, voltmeter readings would be impossible to observe but interest exists generally in operating additional electric circuits with the voltage pattern produced and not in directly observing information so represented.

The number of binary digits that can be represented per second is called the frequency. In the initial consideration of the circuit in Fig. 1. it was assumed that the voltage at point B could change only once per second. A binary digit, therefore, lasts for one second; hence, the frequency would be one. In reducing the time required to represent information, it was assumed that the voltage at point B could change once per 1/100 second. Thus, the frequency would be 100, since 100 binary digits could be represented in one second.

The period of time required to represent a binary digit ($= \frac{1}{\text{frequency}}$) is called a pulse time. A pulse is said to be present if a binary one is represented during a particular pulse time. Thus, the binary number 1011 would be represented electrically, least significant digit first, as pulse, pulse, no pulse, pulse. This is shown graphically in Fig. 3 at a frequency rate of 100 pulses per second. When denoted in this manner, frequency is sometimes called the pulse repetition rate, or the "rep rate."

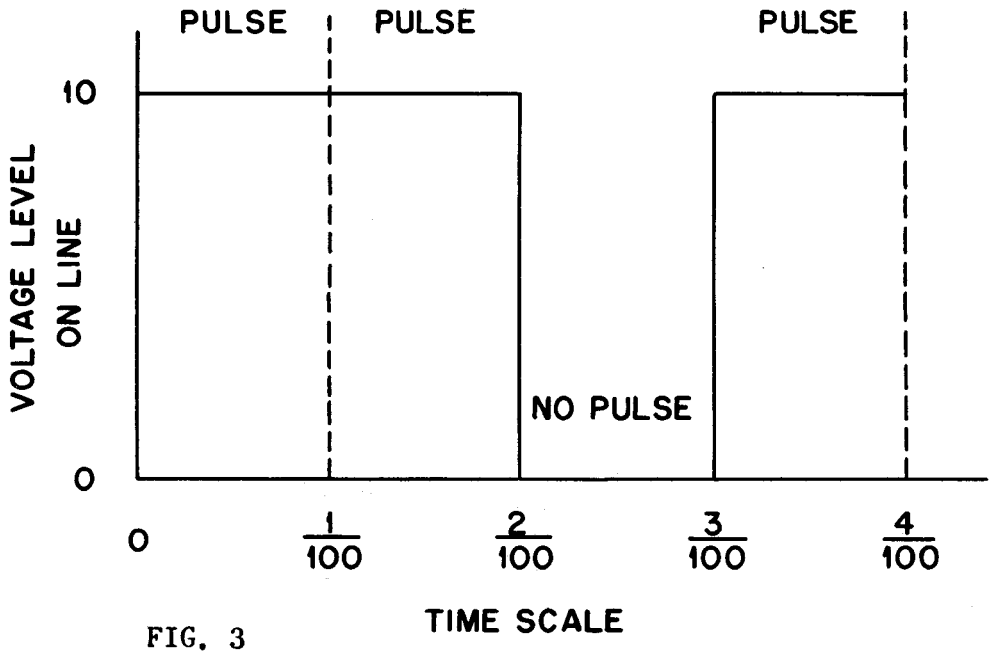


FIG. 3

Voltage changes which persist for relatively long periods of time are called signals. In general, signals will be used for control purposes while pulse patterns are used to represent information.

SECTION 4

Characteristics Of A Simple Computer

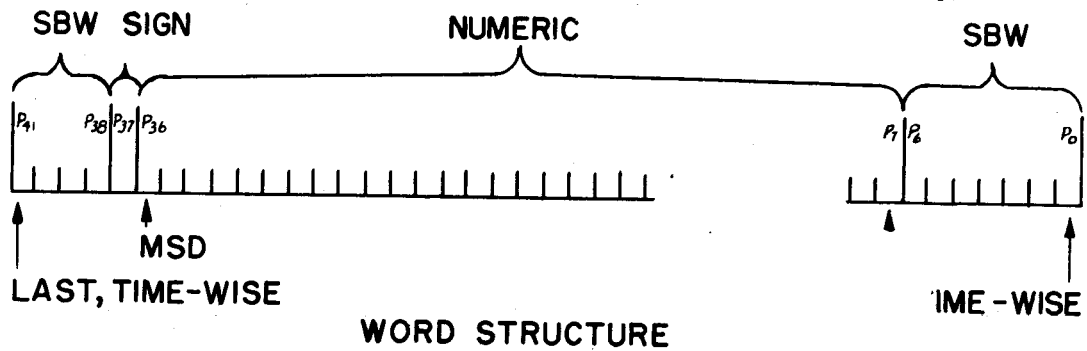
We are now ready to define the characteristics of the computer mentioned in the first paragraph of this chapter. The reader is again reminded that the description to follow is not that of the UNIVAC, but nevertheless helpful in the understanding of the operation of UNIVAC.

In the simple computer to be considered, the "rep rate" is assumed to be 1,000,000 pulses per second or a frequency of 1 megacycle, (1,000,000 cycles per second). The time required to generate a single pulse or to represent a single binary digit will be 1/1,000,000 second or 1 microsecond (1μs). It shall be assumed, further, that thirty binary digits will be sufficient to represent the range of numbers that will be used. In order to standardize the timing of the computer, every number shall be represented by thirty binary digits whether or not all thirty positions are necessary. This basic unit of information is called a computer word or word. It has been shown that only one binary position can be represented at one instant of time; binary one with a high voltage or presence of a pulse or binary zero with a low voltage or absence of a pulse. Hence, the entire word must be represented by a serial pattern of pulses. It is possible to represent the word (with respect to time) with the most significant digit (MSD) first or with the least significant digit (LSD) first. To facilitate arithmetic operation the least significant digit will be represented first followed first followed by succeeding digits in order of significance up to the most significant digit.

An additional pulse position will be added after the MSD to indicate the sign of the quantity. No pulse in the sign position will indicate a negative quantity; a pulse in this position will indicate a positive quantity.

In order to allow a fixed interval of time between successive words on a line, additional pulse positions will be assigned which will never contain pulses. There will be seven such pulse positions before the LSD and four following the sign position and will be called the space between words (SBW.)

The final structure of the word consists of 42 pulse positions. The first of the seven pulse positions preceeding (time wise) the LSD will be designated as P_0. The remaining positions are numbered consecutively through p41 which is the last of the four pulse positions of the SBW following the sign position. This structure is illustrated in Fig. 4.



WORD STRUCTURE

FIG. 4

The basic pulse frequency of 1 Mg. fixes the time to represent one pulse at 1μsec. Since the word contains 42 pulse positions, the time to represent one word, or the word time, is 42 microseconds.

SECTION 5

Component Units Of The Computer

At this point, descriptions of certain component units and certain operations of the computer will be introduced. Specifically, the discussion below will attempt to describe the operations of flip-flops, gating and buffing, the comparator, delay mechanisms, adders, complementing, counting, registers, shifting, distribution and collection, function tables.

Dynamic Versus Static Signals And Storage

A radio message may be thought of as dynamic. During the reception of information some of it has already been given, some of it is immediately available, while still more is yet to arrive. In contrast, the message once received and written down is static: The message may be examined in whole or in part at any time after reception.

If the radio message were to be repeated continuously, then the information contained in the message would be dynamically remembered. Any one desiring the information would merely tune in at any time and listen once during through the entire message. (Obviously, the message would only be received in its correct arrangement if the listener tuned in just at the beginning.)

Much of the information in the computer is in dynamic form; it circulates around loops containing electrical delays. Such information can be observed serially as it passes some point in the path, yet all the information is not available simultaneously.

In contrast there are static storage devices in the computer which retain information in static form. All the information stored in static form is immediately available, but it may be realized only by sampling or probing the static device with some signal in order to learn its present state.

One of the most common forms of static memory is the flip-flop (FF). It is indicated on block diagrams by the symbol:

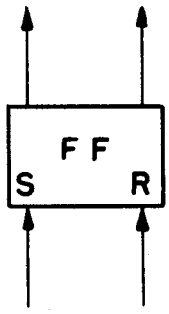


FIG. 5

It is a memory for one binary digit; it has two stable states, one representing zero and the other representing one. These two states are indicated by S (= set) and R (= restore). Its use is sufficiently broad that the binary notation is not always appropriate. For example, it can, in a binary computer, store the sign digit, that is, either a + (which is equivalent to a one) or a - (which is equivalent to a zero). The flip-flop is also useful for converting from a pulse to a static signal. For example, one pulse may indicate when a static signal is to start and an other pulse when it is to stop. The FF can be used for generating such a static signal. The duration of the signal is fixed by the interval between pulses.

When a pulse is applied to the "set" input, the FF is said to be set; when a pulse is applied to the "restore" input, the FF is said to be restored or cleared.

If the flip-flop is in the "set" state, the set output will be at the signal level and the reset output at the no signal level. The opposite is true for the "restore" condition. In some logical circuits, we shall be interested in only one of the outputs. In this case only the necessary output will be shown.

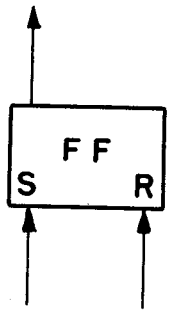


FIG. 6

Of course, the restore output line is still present in this case, but is not shown.

Gating And Buffing

Circuits known as gates are the chief form of switching used in the UNIVAC. As their name implies, gates permit or prohibit passage of signals from one point to another. Gates are indicated by the following symbol:

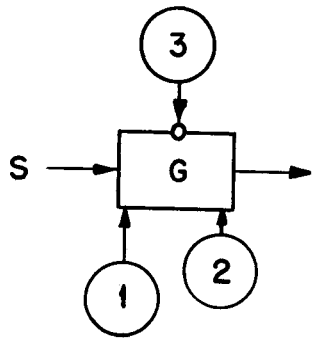


FIG. 7

The signal being gated (S) appears on the left. The various control signals are indicated as 1, 2, and 3. In order for signal S to pass through gate G, signals 1 and 2 must be present and signal 3 must be absent. Any other ar-

rangements of signals is sufficient to prohibit signal S. Signal 3 is often called an inhibiting signal (note the small circle at the point of connection) while signals 1 and 2 are called permissive signals (without the circle connections).

The indications within the large circles always imply the existence of some other device which generates the required gate signals. Typically, another gate can generate such gate signals.

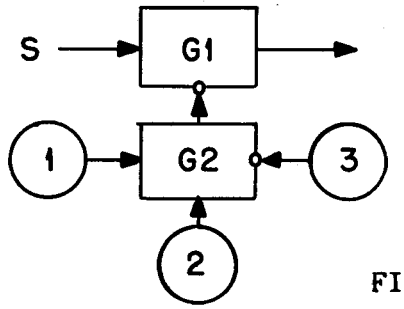


FIG. 8

Signal S can pass through G1 if G2 developes no signal. G2 can only develop a signal if 1 AND 2 are present AND signal 3 is absent. Gate circuits are sometimes called "and" circuits because they require the presence of this AND this signal in order to operate. Gates may be "opened", "activated" or "excited" in order to permit signals to pass. They may also be "inhibited", or "closed", or they may "prohibit" signals from passing.

It is important to realize that every gate passes ONE signal under the influence of OTHER signals. In general, the control signals must overlap in time the signal being gated. The word "alerted" is applied to a gate to describe the condition when one or more but not all of its SEVERAL control signals are present.

A "pulse" signal is one which has a time duration of approximately one pulse time. Longer signals are generally referred to as static type signals.

For example, if a word is passing some point and the sign position is to be examined, then a gate with the desired pulse signal, say p37, is connected to the point.

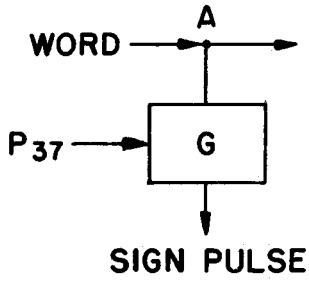


FIG. 9

If a pulse is present at point A during the sign time of a word, it is also gated through G by the p37 pulse. The p37 pulse for the gate is generated elsewhere in the computer.

The converse of gating is buffing. The buffer is indicated by a symbol B. A typical buffing circuit is shown as:

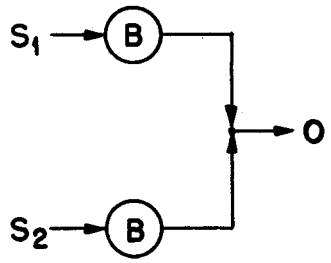


FIG. 10

The purpose of buffing is to combine several signal sources into a single line without interaction among the sources. Thus signal S1 cannot pass into S2 but only through its buffer B to the output O. Either signal S1 or signal S2 can pass into the output O. For this reason the buffing circuit is sometimes called an "or" circuit.

Gating signals can be applied through buffers, thus:

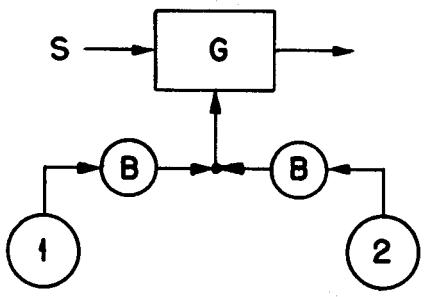


FIG. 11

Signal 1 or signal 2 may open or activate the gate to permit signal S to pass through the gate. At least one signal must be present; when both signals are present no new situation has been created.

It is not necessary to show buffing on the logical diagram if it is understood that such elements exist to prevent back-circuits. It will be assumed that no signal can be passed in the reversal direction of the arrows. Thus figure 11 will be drawn as:

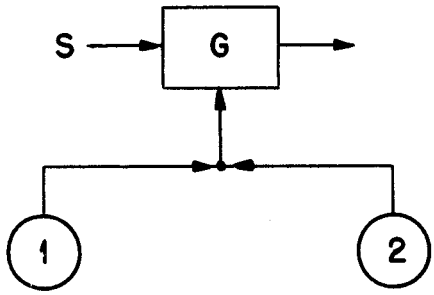


FIG. 12

To determine at some time after p37 if the flip-flop is set or reset and, hence, whether the word was positive or negative we must sample one of the output lines.

If we need the information at time p_1 to operate some other circuit we can sample a gate fed by the flip-flop with a p_1 as pulse as in figure 13.

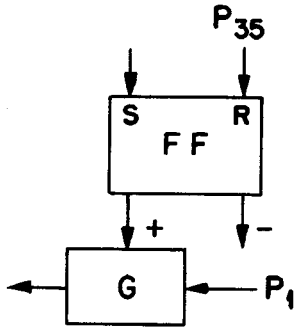


FIG. 13

A combination of gates and flip-flops can illustrate some basic circuits of a computer. A gating arrangement which is tested for the presence of a sign pulse was shown earlier. If the information obtained from the test (or the sampling as it is sometimes called) must be remembered then a circuit like this can be used:

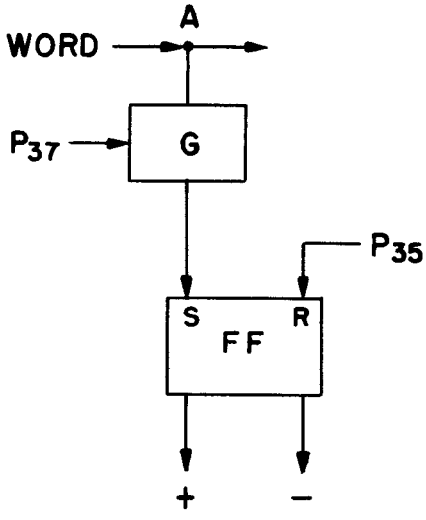


FIG. 14

If a sign pulse is present, it is gated by the p37 pulse to the S-input of the FF. The FF will remain set until the p35 pulse resets it. The p35 occurs two pulse times before the p37 during each word time. Therefore, the behavior of the FF is as follows: at all times the "-" or restored side of the FF is excited except when a sign pulse (plus sign) passes point A. When the sign pulse occurs the FF is set. (If there are 42 pulse times per word then the FF remains set for 40 pulse times each time a sign pulse passes point A.)

The Comparator (Magnitude)

A binary magnitude comparator is a simple combination of gates and flip-flops. Its purpose is to compare the magnitude of two unsigned binary combinations. First, it is important to realize how to compare two binary quantities. Suppose the two quantities are:

A = 0 1 1 0 1 1 0 1 1 1

B = 1 1 1 0 1 0 1 0 0 1,

With the LSD's shown on the right. Begin by comparing the LSD's. They are both one's, therefore, the numbers so far are equal. Compare the second digits: A has a one, B has a zero; therefore, A is now larger. The third comparison is in favor of A and so forth, until the MSD is reached when the decision finally falls to B.

The circuit for this comparison is as follows:

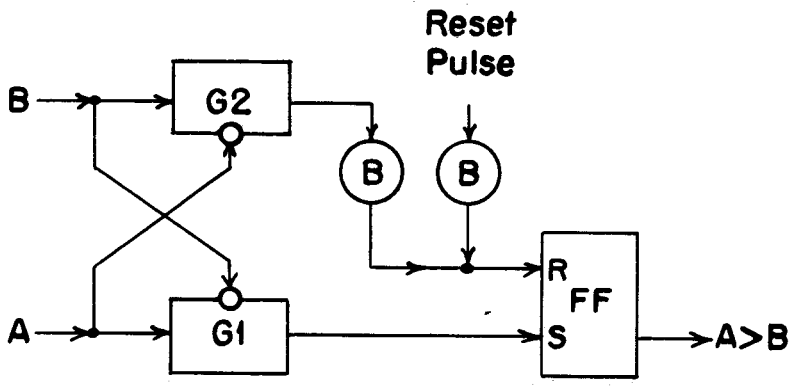


FIG. 15

B is assumed larger than A unless proved other wise; therefore, the restoring pulse removes the A>B before the comparison starts. As the first or LSD pulses of A and B reach their respective inputs, both gates, G1 and G2 are closed by the opposite input. No pulse reaches FF. On the second pair of digits the pulse on A inhibits G2 and passes through G1 to set FF. A now is greater than B. The third pair of digits follows the same procedure. When the fourth pair of digits arrive, the pulse in B inhibits G1 but passes through G2 to restore the FF, and so forth. When the MSD's are compared, the FF is left in restored condition and the fact is determined that, A is not greater than B.

Suppose the two quantities A and B had been equal: no pulses would have reached FF and it would have remained restored. Therefore, the comparator only indicates two conclusions $A>B$ and $A\leq B$; it can not distinguish between $A<B$ and $A = B$.

A more accomplished comparator is shown in Fig. 16.

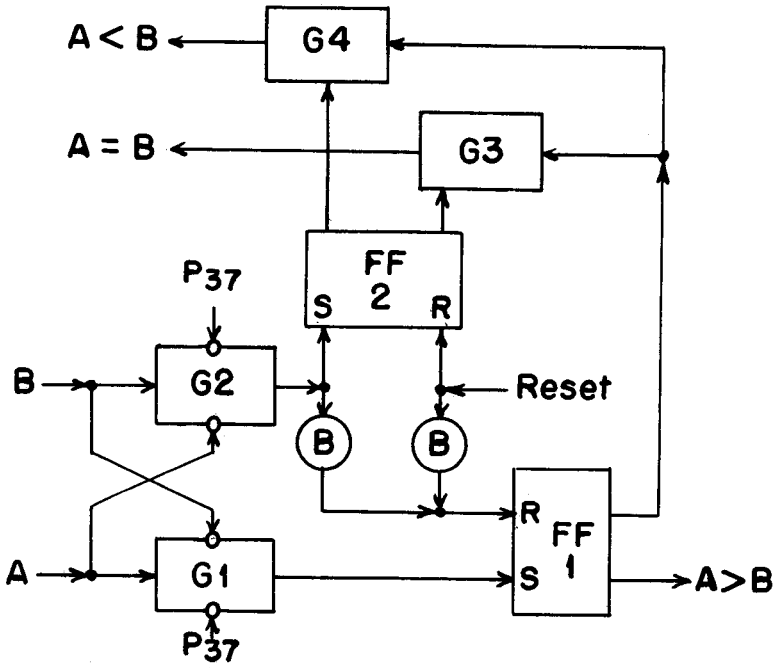


FIG. 16

When the restoring pulse restores FF1 and FF2, G3 is open and the A = B signal is given. This signal persists as long as identical zeros and ones appear at both A and B. If a one appears at input A with a zero at input B then FF1 is set. If the opposite event occurs then FF2 is set and FF1 is restored if it had been previously set and G4 is opened giving an A<B signal. Whatever state exists after the MSD's have been compared indicates the proper conclusion as to the magnitudes of the two quantities, A and B. This sign pulse suppression on G1 and G2 prevents a comparison of sign which can be compared by another type of circuit.

Delay Mechanism

If signals can exist in dynamic form, then an electrical delay constitutes a type of memory or storage facility. The common acoustic echo illustrates this storage phenomenon. For a period of time after generating a sound, the sound is stored in the form of acoustic waves which travel from the sound source toward a reflector. The reflector is unessential to the phenomenon but makes it easier to appreciate the storage principle. Suppose, instead of a reflector, there were some receiving device at a distance from the sound source. The same principle holds true. By the fact that the sound has experienced a delay, that is, requires a finite transit time for travel between source and receiver, it has been stored or remembered for a length of time equal to the transit time. Ordinarily, the transit time for propagation of an electrical effect, through, say, a wire, is much too short to realize any practical or realizable storage capacity. However, there are several known methods by which the propagation of electrical effect can be retarded. One of these is to convert the electrical, effects into acoustic patterns and, thus make use of the much slower rate

of acoustic propagation. The mercury memory with its transducing crystals at each end is typical of this type of storage device. The acoustic propagation through the mercury between the transmitting and receiving crystals is exactly analagous to the acoustic transmission described above.

Another type of delay is the electric delay line. Any coaxial cable with its distributed capacitance and inductance represents a delay line in which transit time for signals is longer than over ordinary wires. A convenient equivalent device requiring less physical space for a given delay time, is a lumped parameter delay line in which coils and capacitor are wired together as follows:

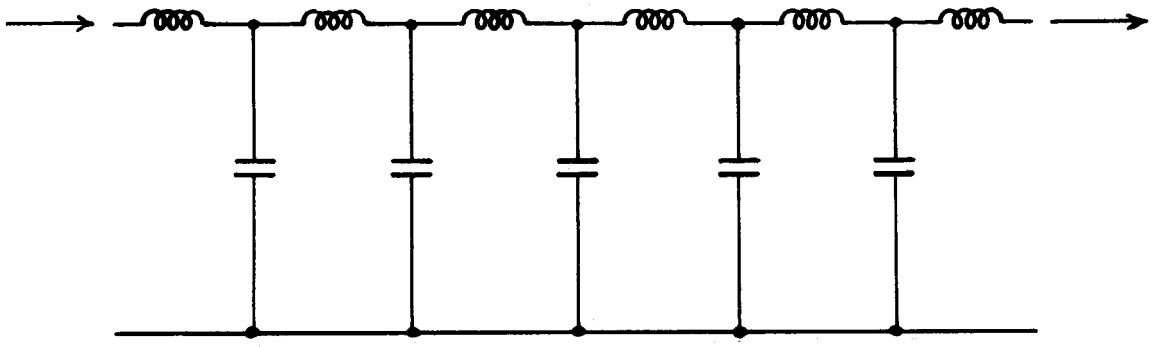


FIG. 17

The total delay of such a line is fixed by the number of sections. It is the pulse rate which determines how much storage such a line represents. The higher the pulse rate, the more pulses can be sent into the line before the first pulse reappears at the opposite end of the line.

The important concept of timing now becomes a matter of inserting and removing the correct amounts of delay in order to bring about the synchronous arrival of various pulses and signals at a given point.

In the logical diagrams, both acoustic and electro-magnetic delays will be represented by boxes containing a number to indicate the delay time in pulse time units. Thus, figure 18 represents the delay of two pulse times.

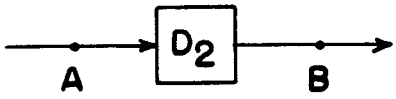


FIG. 18

If a pulse is fed into this line as a p_2, it will control the voltage of point A for the time p_2, but will not control the voltage of point B until time p_4 or two pulse times after its introduction to point A.

Adders

By combining gates, buffers, and delays, several simple types of adders can be constructed. One example is called the half-adder. The half-adder combines the pulses of two words and produces either sum or carry pulses according to the laws of binary arithmetic. These rules are as follows:

0↓0 = 0
0↓1 = 1 0 = 1 (sum pulse)
1↓1 = 0 (sum pulse) and 1 (carry pulse.)

A half adder is shown in figure 19.

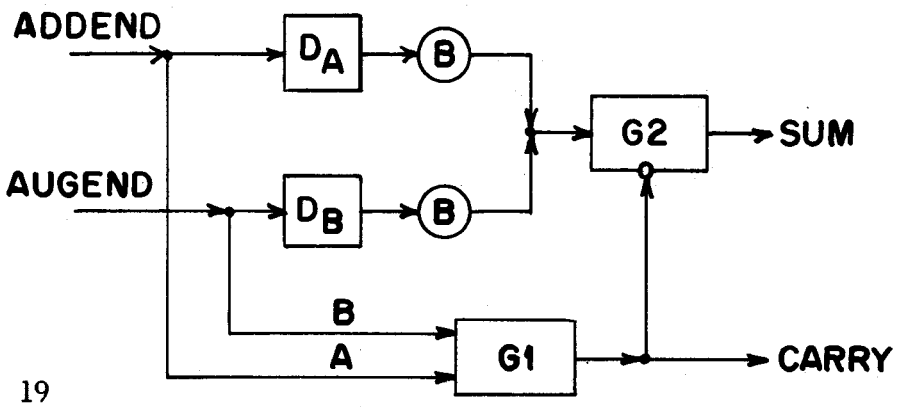


FIG. 19

The two inputs are marked addend and augend. If no pulses enter either input, no pulses are emitted. If a pulse appears on either input, it reaches G1 and also passes through the delay (D), to a buffer to G2. Since G1 had only one input signal it did not open and, therefore, did not inhibit G2. Therefore, a sum pulse was produced.

If two pulses are applied to a half adder, one to each imput simultaneously, then G1 does operate. The two pulses become one pulse in G1. The output of G1 becomes a carry pulse and also inhibits G2. The two input pulses pass through the delays and buffers to G2 but are prevented by G2 from passing to the sum output. The two delays are inserted in the path to G2 in order to delay the pulses for the amount of time required for G1 to develop its inhibition on G2 if both inputs of G1 have been excited. The half adder thus follows all the rules laid down for it above. The half adder will be shown on the logical diagrams by the following symbol:

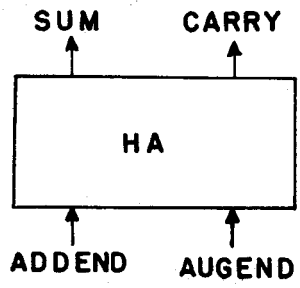


FIG. 20

In some cases the output of the carry line is not used and will be ommitted from the symbol. In a half adder, the missing operation is the transfer of the carry pulse back to the input of the adder so that it can be combined with the next digit pair entering the adder. First, such a device would require a one pulse delay between the carry output and the input of the adder. However, there is still another problem in that the half adder only has two inputs; the carry pulse could not be buffed into one of the two present inputs because the carry pulse would be lost if another pulse should occur as a digit of the word entering that word. Therefore, a third input

must be devised to provide for the triple input conditions when there is a one at both the addend and the augend inputs and also a carry pulse to be added in from the previous addition. For this operation a full binary adder is required. The full binary adder can be constructed from two half adders, which is how the half adder received its name. An arrangement of two half adders which forms a full binary adder is shown in figure 21.

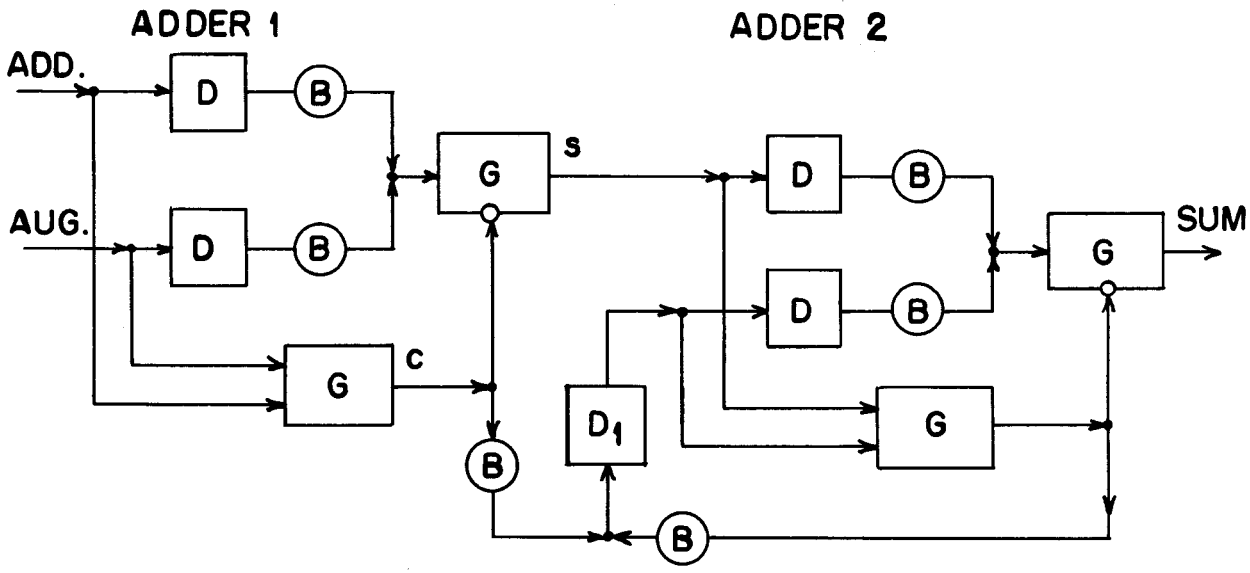


FIG. 21

Adder 1 combines the input digits to form the proper sum. If a sum pulse occurs it passes directly into and through adder 2 to become a final sum pulse. If a carry pulse occurs from adder 1, it enters a one pulse delay, D1, for storage until the next pair of digits are added in adder 1. If they produce a sum pulse from adder 1, then the sum pulse and the carry pulse formed from the previous addition operation are combined in adder 2. This situation, incidentally would produce a carry pulse from adder 2 and no final sum pulse. If the carry pulse occurs when zeros enter the addend and augend input at the next digit time, then the carry pulse passes through adder 2 to become a final sum pulse.

On the logical diagrams a full binary adder will be represented by the following symbol:

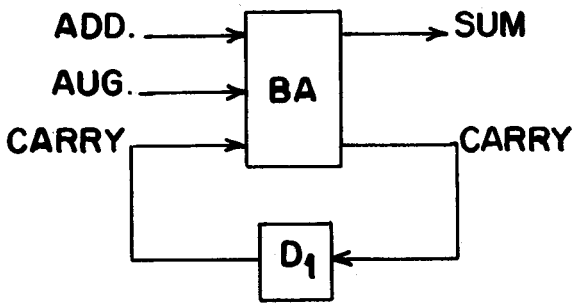


FIG. 22

A typical register for a binary computer is shown in figure 27. The direction of circulation in the register is determined by the polarization of the PFR's and presence of buffers (B). If the word length is 42 pulses then the register must represent exactly 42 pulse times delay. The main delay element (which could be an electric delay line or a mercury acoustic line) is 38 pulse times long. Each pulse former represents 1 pulse time of delay and the compensating delay (Dc) represents 2 pulse delay or a total of 42 pulse times for a complete circulation.

There are three gates, G1, G2 and G3. G1 is the output gate and G3 is the input gate. G2 is called the clear gate. Its function is to interrupt the path of circulation so that any pulses in the register are cleared out. The clear gate is controlled by signal S2; when S2 is present G2 is inhibited and the register is cleared. The duration of S2 is important, for the signal S2 must certainly last at least as long as the time required for a word to pass the clear gate.

The output gate, G1, is controlled by S1. When S1 is present, the pulses passing the output gate connection can be communicated to another register or other elements of the computer. The operation of G1 has no effect on the pulses in the register.

The input gate, G3, is controlled by S3. Usually S2 and S3 occur together because each transfer to a register such as here illustrated, requires that the previous contents of the register be cleared out before the new word enters. Actually the duration and existence of signal S2 and S3 can be identical.

Since G2 and G3 are at the same time point within the register (there being no significant delay between these two gates) the input can be opened at the same time that the clear gate is closed or inhibited. If G2 and G3 are at the same point in time, and the direction of circulation within the register is as shown, then there must be exactly one word delay between G3 and G2 from the standpoint of an entering word. The existence of the buffer prevents the entering or incoming word from reaching G2 except by the long route.

Now, if the two gates are operated together, clearing of the first pulse position of the previous word begins just as the first pulse of the new word passes through input gate G3. Furthermore, G3 should close just after the last pulse of the incoming word. But at the same time, the last pulse of the old word in the register has just reached the clear gate.

Since the first and last pulses of the information portion a word in a register are separated by the space between words (SBW), there are several pulse times available during which to close the input gate and open the clear gate. The clear gate, G2, must certainly open before the first pulse of the new word reaches it. Due to the precise timing throughout the computer, the time relation between the old word in the register and the new word entering a register is identical.

The timewise separation of G1 and G2 by the D2 delay will be explained later.

Shifting

In the ordinary desk calculator, the operation known as shifting is accomplished

by moving the accumulator dials on the carriage with respect to the input keyboard, The shifting operation is required in multiplication and division. Effectively, shifting is simply multiplying a quantity by a power of the base of the number system. Thus a decimal calculator multiplies by powers of 10 (positive and negative) as the carriage is moved to the left or right.

In a binary device a shift of one digit position to the right is effectively multiplying the quantity by 1/2 or 2^{-1}. Shifting left two places is multiplication by 4 or 2^2. The method of shifting in a dynamic circulating register is accomplished by adding or removing unit pulse times of delay. Thus in figure 28, shifting circuits have been added to a register. The normal path of circulation is through gate G5.

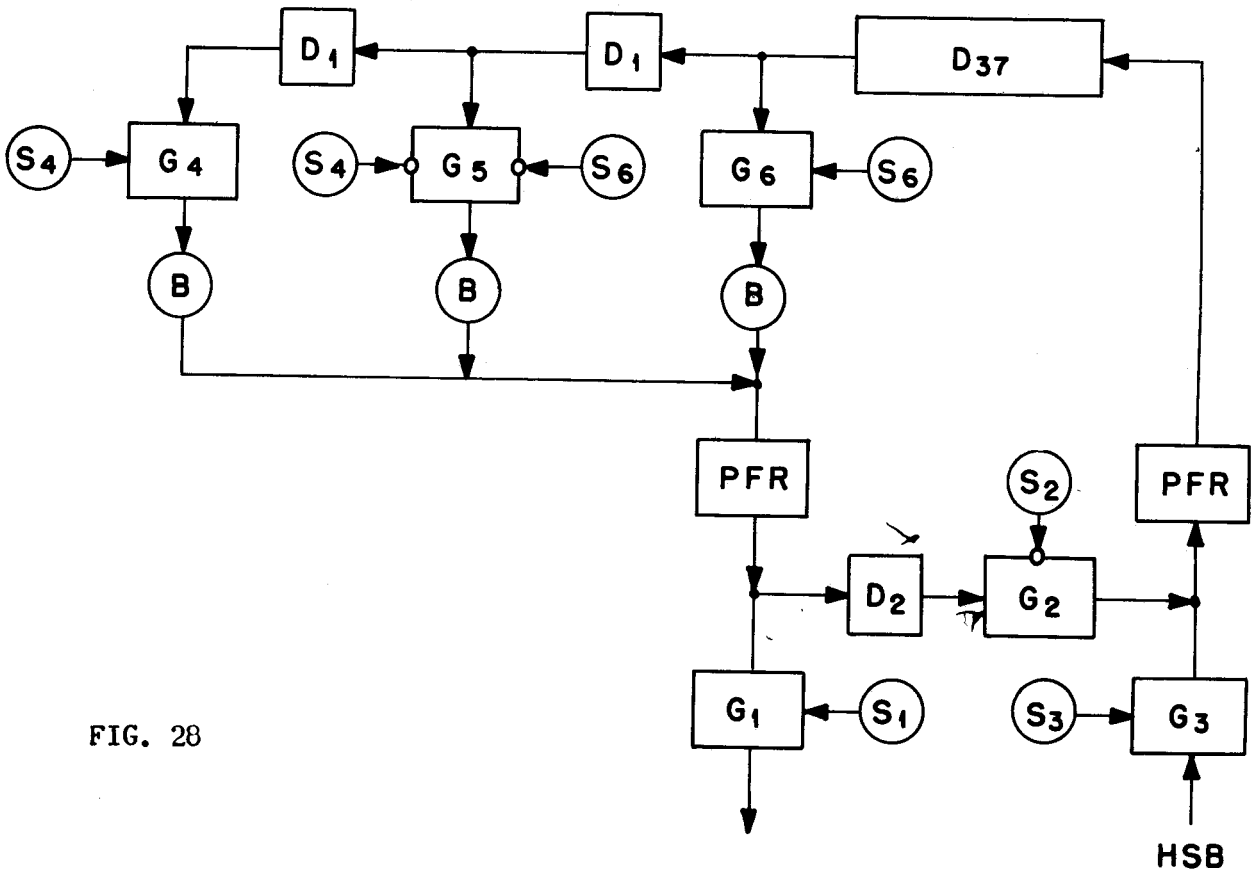


FIG. 28

Distribution And Collection

The flip-flop and binary counter were shown to be a form of static binary memory. The circulating register is a form of dynamic memory. If both types of devices exist within the computer, there must be some means of converting from dynamic to static storage and from static to dynamic storage. The former process is called distribution and the latter is called collection.

Figure 29 shows the schematic arrangement for distribution. First, there is an electrical delay line shown on the left. There is a delay of one pulse time between each tap. The connection from the delay line to the gates are called taps.

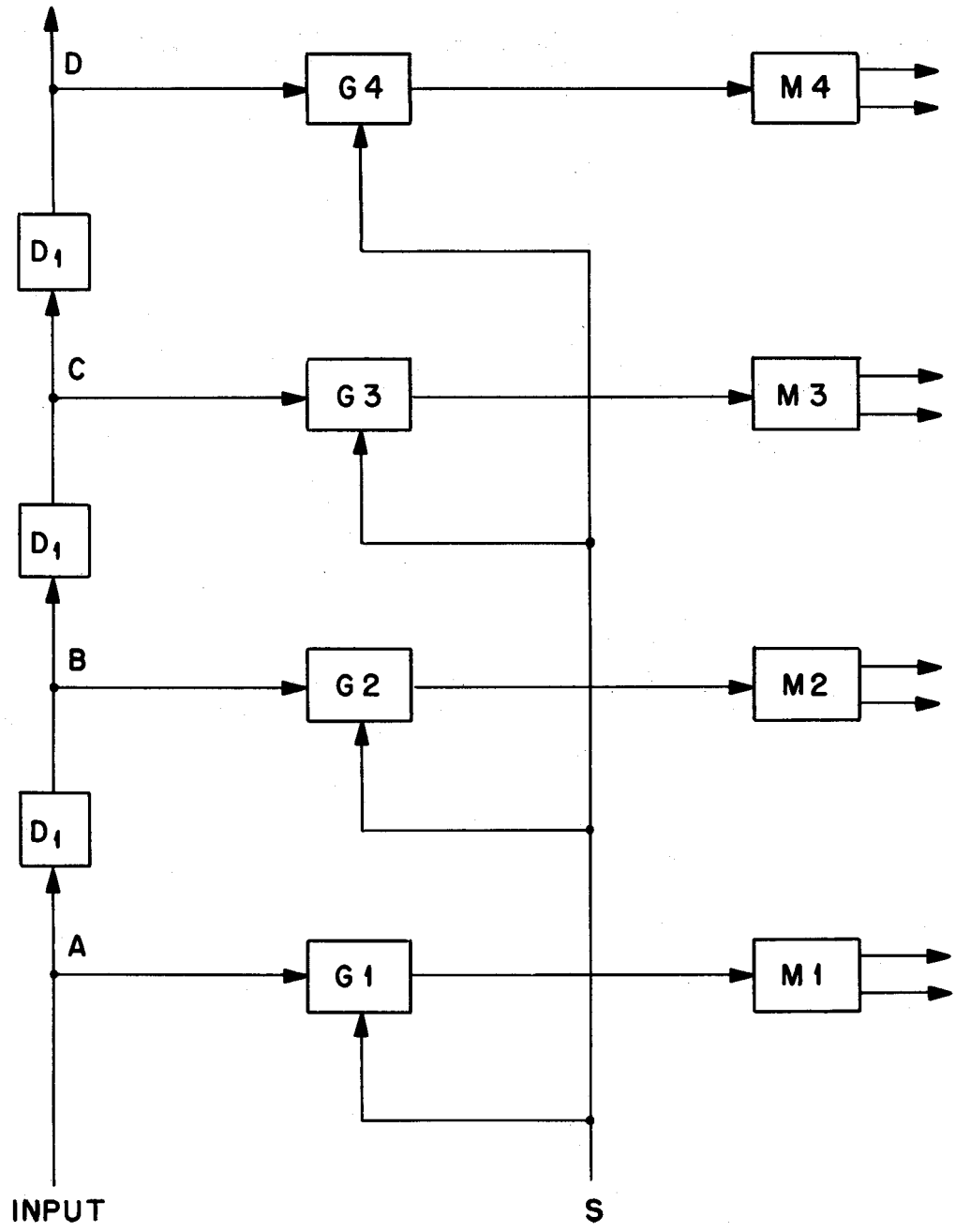


FIG. 29

When a sequence of pulses enters the delay line, the signal S can be so timed that each gate (all gates are sampled at the same instant of time by S) will be operated by the presence of a pulse at its tap. If a pulse exists at any tap, it is transferred through the gate to the elementary binary memory (M). These memories (M), for example, can be flip-flops. They might be binary counters, in which case the special clearing inputs of the BC's would be used and not the stepping inputs.

Suppose we wish to set up the p_7, p_8, p_9, and p_{10}, positions of the computer word in M_4, M_3, M_2 and M_1. P_7 controls the voltage at point A during p_7, but does not control the voltage at point B until time p_8. During p_8 time, p_8 controls the voltage at point A. Two pulse times later at time p_{10}, the p_7 pulse will control the voltage of point D, p_8 the voltage of point C, p_9 the voltage of point B and p_{10} the voltage of point A. Hence, if we sample the gates at p_{10} ($S = p_{10}$), pulses present in any or all of these four positions will be transferred into the desired memory, $M_1 \ldots M_4$. Although the sequence of pulses continues to move through the delay line, no other pulse positions of the computer word can be transferred to the binary memories, since the gates are sampled only during time p_{10}.

Therefore, a word circulating in a register can be converted into static storage by using a distribution line, a gating system and a set of flip-flops or other binary memories. The time length of the delay line must equal the numbers of pulses being converted; the number of gates and binary memories must likewise equal the number of pulses in the sequence.

The process of <u>collection</u> is the opposite of distribution. Figure 30 shows the schematic arrangement for collection. Information is stored in the binary memories M_1, M_2, M_3, and M_4. One output of each memory element is fed to a gate. Signal S samples each gate during successive pulse times, producing an output pulse on the transmission line, for each gate alerted by its memory element.

If the p_{10} position of a word is stored in M_4, p_9 in M_3, p_8 in M_2 and p_7 in M_1, we can convert this to a sequence of pulses by making $S = p_7$. Then G1 is sampled at p_7 and will produce a pulse if the state of M_1 indicates a binary one. G2 is sampled at p_8, G3 at p_9 and G4 at p_{10}.

Only one output of each memory element is used to alert its gate. Hence, a gate will have an output only if the "1" output line of its memory element is excited, even though all gates are sampled by signal S.

Function Tables

A function table (FT) is a device which can decode many input lines into a single output line or encode a single input line into many output lines. The former type function table is called a decoding FT while the latter is called an encoding FT.

Consider the output lines of two flip-flops. The excited not-excited combination of these lines enables us to represent numbers 0, 1, 2, 3, as shown by the following table.

Because of the nature of the decoding function table, which provides the input in figure 32, only one input line can be excited at any instant of time. Thus, if the "00" line is excited, the S_1, S_2 and S_6 lines will be picked up. The "10" line will pick up S_3, S_5 and S_7, but no others. This makes it possible for a single input line to excite a pattern of output signals which will cause the computer to carry out some pre-determined operation.

SECTION 6

Operation of the Computer

The Instruction Code

The computer was designed to carry out only eight instructions, in order to keep the logical circuits as simple as possible. Therefore, only the p30 to p36 positions of a word are decoded as an instruction. The p30 to p33 positions indicate the address of a memory location if the memory is to be involved in the instruction. Otherwise, these positions are not used by the computer. When a word from the memory is needed, the p32, p33 position indicate which of the four within the selected channel is to be read out. Thus, an address of 1000 would indicate the zero word of channel two. Or if we number the words from 0 (for the zero word of the zero channel) through 15 (for the 3 word channel of channel 3), 1000 would indicate word 8 of the memory.

The eight possible pulse combinations of the p30 to p36 position gives the code for the eight instructions the computer will execute.

Figure 33 indicates the portion of the word which the computer can use in instruction.

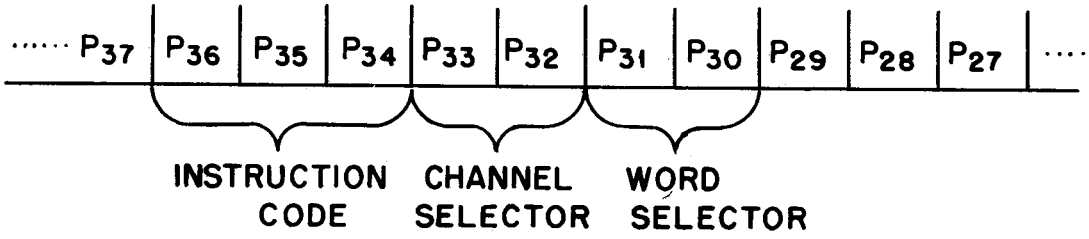


FIG. 33

The code is described as follows:

Instruction	Description
000m	Transfer the contents of the memory location indicated m to rA, clearing rA of its former contents and leaving (m) unaltered.
001m	Transfer the contents of rA to the memory location indicated, clearing m of its former contents and leaving rA unaltered.
010 0000	Transfer the contents of rA to rB, clearing rB of its former contents and leaving rA unaltered.

011m	Transfer the address m of the instruction word from the static register to CC, clearing CC of its former contents.
100m	Transfer the contents of memory location m to rB clearing rB of its former contents; transfer (rB) and (rA) to the adder and return the sum to rA,; clearing rA of its former contents and leaving rB unchanged.
101m	Transfer (rA) and (rB) to the comparator leaving the contents of both registers unchanged: if (rA) > (rB) transfer the address m of the instruction word from the static register CC, clearing CC of its former contents, if rA $\leq$ rB do not change CC.
110 0000	Shift (rA) one digit position to the right replacing the sign position with binary zero.
111 0000	Shift (rA) one digit position to the left replacing the LSD with binary zero.

Organization of the Computer -The overall layout of the computer is shown in figure 34.

Memory - The main memory consists of four acoustic delay lines each capable of storing four words, or a total storage of sixteen words. These words can be instructions or data.

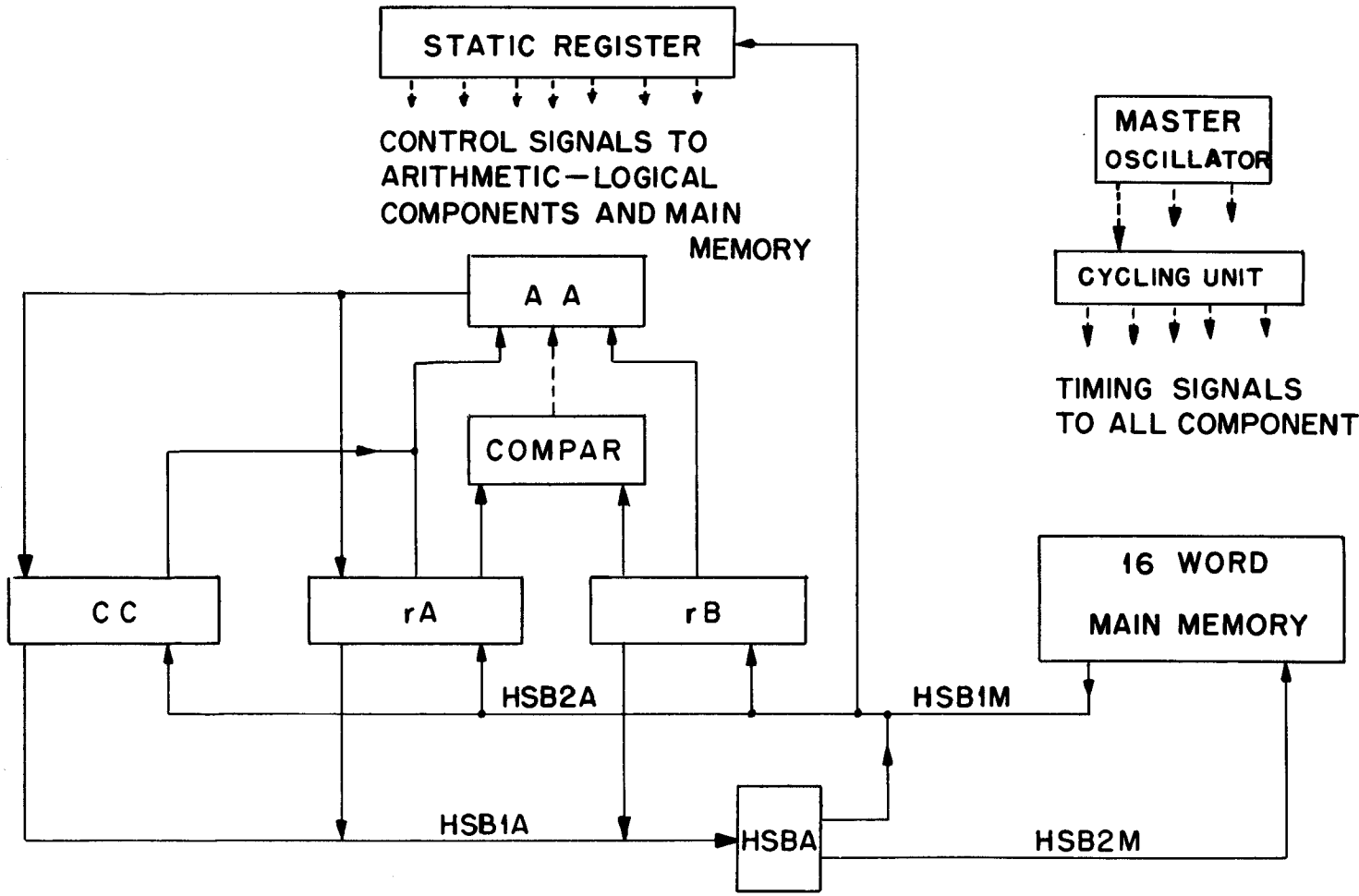
All instructions are stored in the memory as numbers, and are distinguished from data only by the manner in which the computer makes use of them. There is no section of the memory specifically allocated for instructions; data or instructions can be stored in any memory location.

It should be clearly understood that no arithmetic or logical operations are carried out in the memory. It is used only for storage. When a word stored in the memory is required as an operand or for decoding as an instruction, it must be transferred to the arithmetic circuits or control circuits.

The transmission line which carries information from the memory is called high speed bus (HSB)1M. The line which transmits information to the memory is HSB2M.

Arithmetic Circuits - These circuits consist of the algebraic adder (AA), the comparator (CP), and two one-word acoustic type storages called register A, rA, and register B, rB. The latter serve as temporary storages for words to be used by the adder or comparator.

The comparator is used on the 101m order to compare the absolute values of rA and rB, and initiates one of two possible sequences of instructions based on (rA) > (rB) or (rA) $\leq$ (rB). The comparator is also used on the 100m order to determine whether addition or subtraction is to be performed by AA.



BLOCK DIAGRAM OF A SIMPLE BINARY COMPUTER

FIG. 34

The adder is used on the 100m order to add or subtract the quantities stored in rA and rB, returning the sum to rA. It is also used by the control circuits to increase the word stored in the control counter (CC).

Control Circuits - The control circuits consist of the static register (SR), the cycling unit (CU), and the control counter (CC). The static register converts the p30....p36 positions of a word to be used as an instruction into flip-flop storage. These flip-flops drive function tables to produce the necessary signals to carry out the instruction.

The cycling unit generates timing signals which are sent to all components of the computer to synchronize its operations.

The control counter is a one-word, mercury delay line register. Its purpose is to store the address of the next instruction word. The numerical value stored in CC is referred to as the CC-reading. Normally, the instructions are performed according to the numerical value of the memory locations in which they are stored. Thus, if CC initially reads zero, the computer is referred to 0000 for the first instruction word. As the reference to 0000 is completed, the CC-reading is advanced to 0001 and so forth. Hence, CC functions as the sequencing mechanism of the computer.

If the normal sequence of instructions is to be changed, then the CC-reading must be altered. Both and 011m and 101m orders accomplish this. (See Instruction Code).

The transmission line which carries information to rA, rB, CC and SR is HSB2A. HSB1A receives information from rA, rB or CC and transmits it to the high speed bus amplifier (HSBA). This is primarily a switching central and can send information from HSB1A to HSB2A (for a register to register transfer) or to HSB2M (for a register to memory transfer).

Timing - Cycling Unit Signals

We have assumed a pulse rate of 1 Mg. for the computer. Pulses are produced at this frequency by a crystal controlled master oscillator, which maintains the pulse frequency with negligible variance. The pulse output of the master oscillator (one per μ sec.) is not used to represent information directly, but controls all pulse formers in the computer to limit their output time for one information pulse to exactly one microsecond.

The propagation of information pulses through the various components of the computer will tend to distort them, but pulse formers are inserted sufficiently often in the circuits to reshape them and keep their time duration constant.

For control purposes, it is necessary to have signals which occur at less than the basic pulse frequency of the computer. It is necessary, for example, to have a signal to mark the beginning of each minor cycle (word time) to indicate when p_0 position of a word in any of the one-word registers is available for read-out. Such signals will be produced by the cycling unit shown in figure 35. This is nothing more than a one-word register with taps needed for each signal.

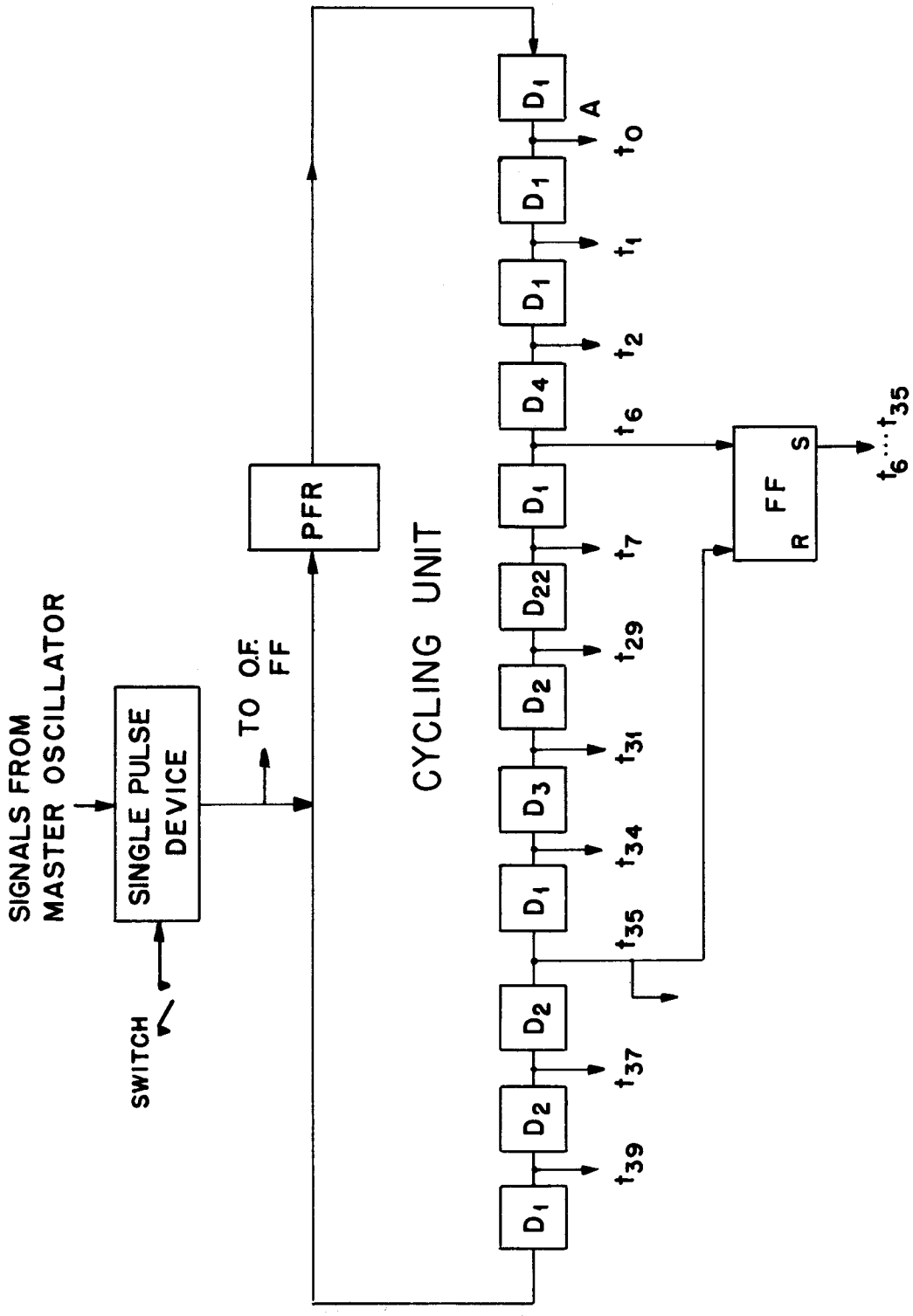

SIGNALS FROM MASTER OSCILLATOR
SINGLE PULSE DEVICE
SWITCH
TO O.F. FF
PFR
CYCLING UNIT
D1
D1
D1
D4
D1
D22
D2
D3
D1
D2
D2
D1
A
t0
t1
t2
t6
t7
t29
t31
t34
t35
t37
t39
FF
S
R
t6 ··· t35

FIG. 35

After the master oscillator is started, the single pulse device is switch operated. This produces one pulse whose duration is fixed at one microsecond by successive pulses from the master oscillator. The single pulse is fed into the recirculation loop of the cycling unit. This passes a PFR and a D1 delay and then produces a signal at tap A for one pulse time. The signal on this line is called a t_0 and indicates the beginning of a minor cycle. After 42 μsec. or one minor cycle, the pulse once more reaches tap A and the t_0 signal is repeated.

Additional signals are needed at various time within the minor cycle and these are produced by tapping off the recirculation loop an appropriate amount of delay after the t_0.

For timing signals which last more than a pulse time, but which repeat each minor cycle, flip-flops controlled by CU signals can be used. Thus, a signal is needed which lasts from t_6 through t_{35} of each minor cycle. This is produced by using a t_6 to set a flip-flop and a t_{35} to reset it. Thus, the signal condition of the set output of the flip-flop is present only during the required time in each minor cycle.

Memory Timing

The main memory of the computer consists of four four-word registers or channels. The channels are numbered 00, 01, 10 and 11. Channel 00 is shown in full on Chart A while only the input and output lines of the other channels are indicated.

Each channel consists of a recirculation loop, a read-in gate, G_1, a read-out gate, G_0; a clear gate G_c; a control gate G_t. The recirculation loop contains 4 x 42μsec. = 168μsec. of delay. Thus, a particular word in a channel is available at the read-out gate once in each four minor cycles. The time for the transit of a word from the read-out gate, through the recirculation loop and both to the read-out gate (four minor cycles) is called a major cycle.

It is beyond the scope of this description to deal with the problem of getting information into and out of the computer. It will be assumed that all memory channels have been filled with information in such a manner that the p_0 position of a word is available at the read-out gate, G_0, at time t_0 is indicated by the cycling unit. This "pulse position-time" reference will be indicated on drawings by indicating at the appropriate point in the circuit (in this case, G_0,) that $p_0 = t_0$. The means that the p_0 position of any word in the recirculation loop will arrive at G_0 at t_0 of some minor cycle. It also indicates that the p_1 position is available at: t_1, p36 at t_{36}, etc.

This reference timing does not necessarily mean that the p_0 pulse will always be the first read-out. If G_0 did not become permissive until t_3, the reference timing would indicate that the p_3 pulse is the first to pass through the gate.

It is now possible to explain the D2 delay in the recirculation path of the one-word register, which separates their read-in and read-out gates. The read-out time from the memory to the transmission line HSB1M has been fixed at $p_0 = t_0$. It is necessary to have a PFR in this transmission line, hence, the time on arrival at HSB2A which feeds the input gates of all one-word registers

is $p_o = t_o$. Thus, if a p_o pulse passes a memory read-out gate at time t_o as indicated by the cycling unit, it does not arrive at the input gate of a register until time t_1 as indicated by the cycling unit due to the one pulse delay inherent in the PFR.

There is no delay, however, between the G_o, and G_1 in the memory channels, hence, information passing from a one-word register to a memory channel must arrive with a reference timing identical with the read-out timing, $p_o = t_o$. But it is necessary to have a PFR in the transmission line, HSB2A, which received information from the one-word registers. In order for a p_o pluse to arrive on HSB2M with the proper timing $p_o = t_o$ it is necessary to start it from the register a pulse time early in order to compensate for the delay of the PFR or at time $p_o = t_{41}$. The D2 delay thus separates the read-in gate and read-out gates of the one-word register by the necessary two pulse times.

Switching Time

Most of the gates in the computer are controlled by signals from the encoding function table. The time required for the FT output signals to reach their new levels following any change in the FT input is called switching time. The switching time allowed for the encoding FT signals is one minor cycle even though they reach equilibrium in less time. The control signals on certain gates must only change during the time the SBW is applied to the gates. A special flip-flop controls all such gates. This flip-flop is called the time-out flip-flop (FFTO) and is always set whenever the input to the encoding FT is changed. FFTO is always set by a t_o signal from the cycling unit and reset by the following t_o, hence, it established the one minor cycle switching time.

The set output of FFTO is inhibitory on most gates, so although FT signals may alert a gate during the time-out minor cycle, the gate is not able to pass information until FFTO is reset by the t_o cycling unit pulse. This ensures that information will always leave a register LSD first.

The Three Stage Cycle Of Operation

There are three logical steps necessary for the computer to carry out an instruction. All instructions are stored in the main memory; hence, the computer first obtains the address of the next instruction to be executed. Second, it makes use of this address to obtain the proper word from the memory and stores the $p_{30} \cdots p_{36}$ portion in a static memory. Third, the output of the static memory is interpreted by a function table which develops a pattern of signals (unique for each instruction) to enable the computer to carry out the indicated operation. This three stage cycle is repeated for each order executed. The three stages of this basic cycle are called by the Greek letters α, β, γ,. The cycle counter CY, a two-stage binary counter, which counts 00, 01, 10 and 10 and then resets to 00, is used to remember which stage of the basic cycle the computer is on. See chart B.

CY - 2	CY - 1
0	0 = α
0	1 = β
1	0 = γ
0	0 = α

Thus, CY provides the input to a decoding function table whose output lines are the α line, β line, and γ line. Of course, only one output line is activated at a time; which one is picked up depends upon the reading of CY.

Description of the Basic Cycle of Operation

Alpha time: The memory address of the next instruction to be executed is stored in dynamic form in the control counter (CC). This register stores a full computer word (42 pulse positions), although only $p_{30}...p_{36}$ are used to store the address. Since only one pulse position of a word is available at any instant of time on the recirculation line of a register, it is necessary to transfer the word in CC to a distributor line and convert the desired pulse positions into static (flip-flop) storage. When this is done, these pulse positions are available for as long as needed and can be used to drive function tables which will in turn develope signals to extract the proper word from the memory. Hence, the operation consists of transferring the word in CC to a distributor line (without clearing CC) and converting the $p_{30}...p_{36}$ pulses to static storage. The seven flip-flop (FF30...FF36) are called the static register (SR).

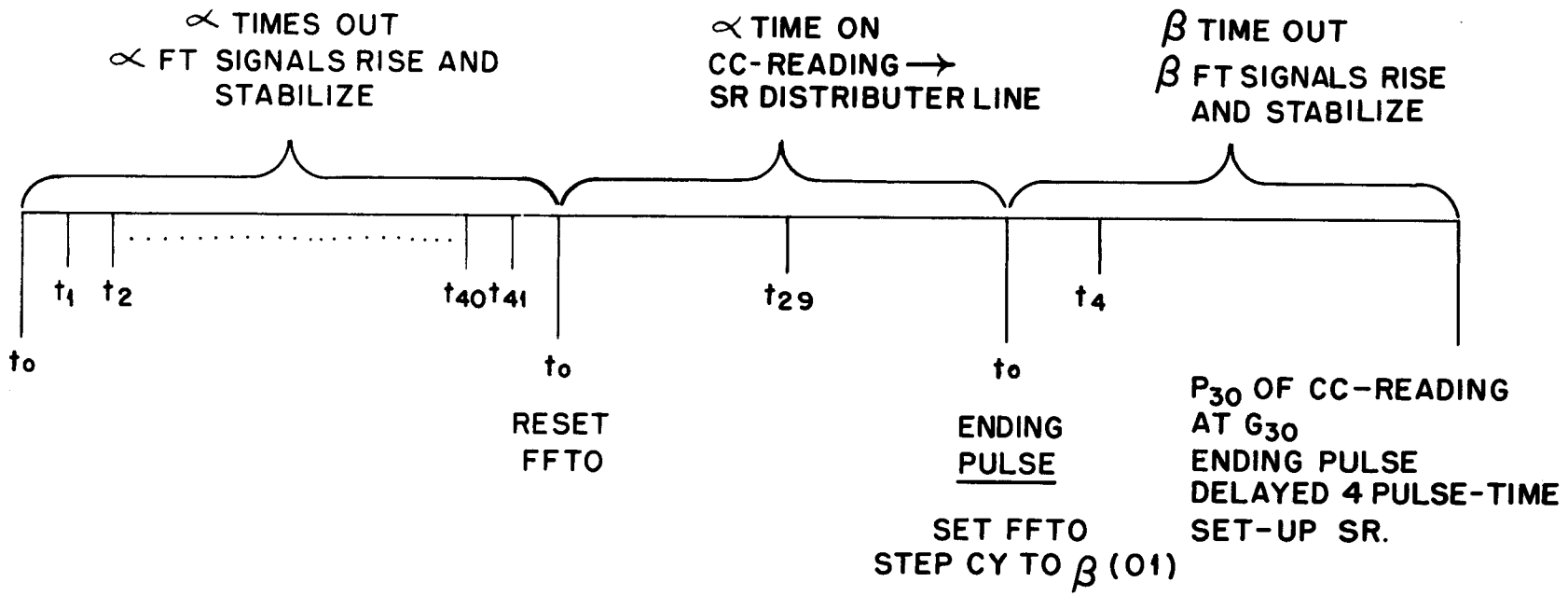
When CY - 1 and CY - 2 are both in the zero state, the α line of the cycle counter decoding function table is picked up. This in turn enters the main encoding function table to select the 8, 9 and 10 lines. These signals are sufficient to effect the transfer from CC to SR.

The minor cycle in which CY reads 00 and FFTO is set is the α TO minor cycle. (See figure 36) It is during this period that FT signals 8, 9 and 10 are picked up and stabilized. By the end of TO, the α FT signals are alerting all gates to which they are applied. A t_o passes gate 25 which is alerted by the set output of FFTO to reset the flip-flop. The time-out inhibition is now removed from all gates. FT signal 8 alerts the read-out gate of CC, G8, to allow the word contained in the register to pass on to HSB1A. The word began passing G8 as soon as FT signal 8 became permissive, but was prevented from reaching any other component by the TO inhibition in the high speed bus amplifier (HSBA). HSBA consists of the PFR, G9A and G9b and the D1 delay in the output of G9A. FT signal 9 is present and when FFTO is reset G9A is alerted while G9B is blocked. Hence, the CC reading is switched from HSB1A to HSB2A.

The word left CC with the reference timing $p_o = t_{41}$, but the HSBA inserts the necessary delay to allow it to enter HSB2A with the proper reference timing $p_o = t_1$. It passes from HSB2A into the distributor line of the static register.

It is now necessary to consider at what time p_{30} will be applied to G30 which controls FF30. When p_{30} is applied to this gate, p_{31} will certainly be applied to G31 gate, etc., hence, the portion of the word that we wish to convert to static storage will be contained in the distributor line, when p_{30} is applied to G30 and it is at this time that the series of gates, G30, G31...G36 must be sampled.

The reference timing on HSB2A which feeds the SR distributor is $p_o = t_1$. If a p_o enters the delay line at t_1 then p_{30} enters at t_{31}. The p_{30} pulse will be applied to G30 fifteen pulses times later (after passing the delays of the distributor line) or at t_{46} or time t_4 of the next minor cycle.



TIMING DIAGRAMS FOR ∝-TIME

FIG. 36

Ft signal 10 is also present during time-on (T.ON) and alerts G10. G10 has the same t_0 cycling unit signal as G25, in the TO circuit but the t_0 which resets FFTO at the beginning of T.ON does not pass G10 even though FT signal 10 is permissive, since FFTO does not change state quickly enough to remove the inhibition on G10 before time t_1. Hence, it is a t_0 one minor cycle later which passes G10. The t_0 pulse output of this gate is called the ending pulse (EP) since it occurs at the end of each state of computer operation. This EP steps CY to 01 and sets FFTO and the computer is now in β TO. It clears the program counter (PC) to zero. (The use of this counter will be explained later). It resets all of the static register flip-flops clearing any information previously set-up in static storage.

The EP also enters D4 delay and emerges at time t_4 of the β TO cycle. But this is the minor cycle following T.ON and it has been shown that it is at this time that the $p_{30}...p_{36}$ pulses of the CC reading are contained in the distributor line. Hence, the EP delayed four pulse times sample G30A, G31A....G36A at t_4 to set-up the memory address from CC in the static register.

Beta time: On β time the next instruction word is transferred from the memory to the static register. The task of reading into or from a memory channel is more complex than reading into our out of a one-word register such as CC. Since the recirculation time for a word in CC is one minor cycle, the word is available for read-out in every minor cycle. However, the recirculation time for a word in a memory channel is four minor cycles, and is available for read-out in only one out of each four minor cycles. Hence, to obtain a particular word from the memory necessitates, first, selecting the channel in which the word is stored and, second, selecting the minor cycle in which the desired word is passing the read-out and read-in gates. The former, a positional reference, is called channel selection and the latter, a time reference, is called the time selection.

Channel Selection - The first two digits of the memory address are stored in FF32 and FF33 of the static register. The output lines of these flip-flops drive a decoding function table which excites one of the lines C_0, C_1, C_2 or C_3. The excited line alerts the control gate, GT of the channel indicated by the first two digits of the address. The channel selector signal will be present, of course, a few μ secs. after the address is set up in SR.

Time Selection - The time selection counter (TSC) is a two-stage binary counter which is stepped once per minor cycle by a t_2 from the cycling unit. The reading of TSC at any instant of time indicates which word is passing the read-out gate of a memory channel. It must be remembered that information in all memory channels circulates synchronously. Thus, when TSC reads 01 it indicates that the 01 word in all memory channels (that is, words 0001, 0101, 1001 and 1101) are passing the read-out gates of their respective channels.

To select the proper minor cycle for reading into or out of a memory channel, it is necessary to compare the two least significant digits of the address stored in FF30 and FF31 of SR with the TSC reading. And when the two agree, to produce a signal for one minor cycle which will alert all control gates, or GT, in the memory.

The comparison of the word number with the TSC is accomplished by gates, G70, G71, G72, and G73. As long as the word number stored in SR does not agree with TSC, there will be a signal output from one or more of the comparison gates. However, when coincidence is reached, there will be no output from any of the gates. The buffed output line of the comparison gates is sampled by means of gate, G1A, to determine when the no signal condition, representing coincidence of TSC and the word number stored in FF30 and FF31, has been reached.

Gate, G1A, is sampled each minor cycle by a t_o from the cycling unit, but the sampling pulse will pass G1A only when the inhibition from the comparison gate is absent. Hence, the t_o will pass only when TSC agrees with the word number. This sampling is effective only on operations involving the memory, since only on these operations will FT signal 1 be present, which is necessary to alert G1A.

When a t_o passes gate G1A, it sets the time selection flip-flop (FFTS) to produce the time selection signal (TS) for one minor cycle. The TS signal is limited to one minor cycle, because the TS signal alerts gate G1B, which passes the following t_o to reset FFTS.

The TS signal alerts all control gates G_t, in the memory channel and in addition is applied to gates G3 and G5. If FT signal 3 is present, TS passes G3 to become the TS signal which is used on reading out of the memory. If FT signal 5 is present, TS passes G_5 to become the TS_1 signal used on reading into memory.

Although the TS channel signal is applied to the control gates of all memory channels, only one gate G_t will develop an output signal because only one of these gates will have a channel selector signal. Thus, if the address of the desired word is 1001 only the control gate of channel 10 will develop an output during the TS minor cycle.

The output of this control gate will alert the read-out gate G_o, the read-in gate G_1 and the clear control gate G_r of channel 10. If the operation is to transfer a word from the memory, the TS_o signal will be present and gate G_o will be fully alerted. This permits one word to leave the memory channel and enter HSB1M. If the operation is a transfer to the memory, TS_1 will be present and G_1 and G_c will be fully alerted. This permits one word to enter the memory channel from HSB2M through G_1 and blocks the recirculation path of the channel for one word time by inhibiting the clear gate, G_c, with the output of G_r.

It should be noted on operations involving the memory, that there may be several minor cycles of time-on during which the computer does nothing but wait for agreement of the TSC and the word number stored in the SR. These minor cycles of time-on are called latency time.

It is now possible to consider the operation of the computer on β time. The ending pulse produced at the conclusion of α time stepped CY to 01 (β) and set FFTO. During this β time out minor cycle, the cycle counter decoding function table decodes the 01 output of CY and excites the β line. This in turn enters the main encoding function table to select FT signals 1, 3, 7 and 11. Also on this TO minor cycle, the channel selector signal is picked up according to the most significant digit of the address stored in FF32 and FF33 of the static register.

On β time-on the TS signal is produced as soon as TSC agrees with the word number stored in SR because FT signal 1 is present. FT signal 3 is also present and the TS_0 signal is produced as well. These two signals are sufficient to read the desired word from the memory to HSB1M. The word leaves the memory with the reference timing $p_0 = t_0$ and after passing the PFR in HSB1M reaches HSB2A with the timing $p_0 = t_1$. From HSB2A the word is fed into the SR distributor line. (It is also sent to the read-in gates of the one-word register, but this is trivial since none of these gates are open.) It is the p30...p36 position of the word (the instruction portion) which are to be converted to static storage. The set-up time then is the same as for α time or at time t_4 following the time selection minor cycle. The ending pulse for the β operation occurs at the end of the β time selection minor cycle. It is obtained through gate G_7. This gate requires FT signal 7, present on the β operation, and the TS signal. The TS signal is necessary because the duration of β time-on varies depending upon the number of minor cycles of latency time. A t_0 passes this gate at the end of the β time selection minor cycle and steps the cycle counter to γ (10), sets FFTO, and resets the SR flip-flops. Delayed four pulse times, it samples gates G30....G36 or a t_4 and converts the p30....p36 position of the word obtained from the memory to static storage for use on γ time.

During β time selection, simultaneous with the transfer of the next instruction word from the memory to SR, the control counter reading is sent to the adder, advanced by 1 in the p30 position and returned to CC.

Gate G11 is opened during the β time selection minor cycle by the presence of FT signal 11 and the TS signal. The CC reading is transferred through G11D, a PFR and gate G58A to become one input to the adder. Gate G58A is open since neither the S_2 nor TO signals are present. The other input to the adder is a pulse (binary one) during the time the p30 position of CC is entering the adder. This is obtained through Gate G11E which is alerted by FT signal 11 and TS, by passing a t_{29} from the cycling unit. The reference timing the CC input to the adder is $p_0 = t_{41}$; hence p30 = t_{29} passed by G11E is added into the p30 position of CC. The result of this operation is to advance by one the address stored in p30...p33 of CC.

The sum which is the new CC reading is returned to CC through a PFR and D1 on the sum output line of the adder and through read-in gate G11B of CC which is opened by FT signal 11 and TS. These same signals operate gate G11A where the output inhibits clear gate G11C in the recirculation path of CC to clear the register of its former contents.

Thus, at the end of a current β cycle, the address stored by CC is that which is to be used on the next α cycle.

Gamma time: On γ time the computer executes the instruction that was brought from the memory and converted to static storage during β time. The instruction code stored in FF34, FF35, and FF36 drive a decoding function table to pick up one of the eight possible instruction lines, which feed the main encoding function table. The pattern of signals produced by an instruction line and the address (if the memory is involved) are sufficient to carry out the indicated operation.

The instruction lines can be excited only on γ time since all instruction lines require the γ line from the cycle counter decoding function table.

The add Order, 100 m, covers most parts of the computer and will thus serve as an excellent illustrative order for γ time. The 100 m order is executed in three steps:

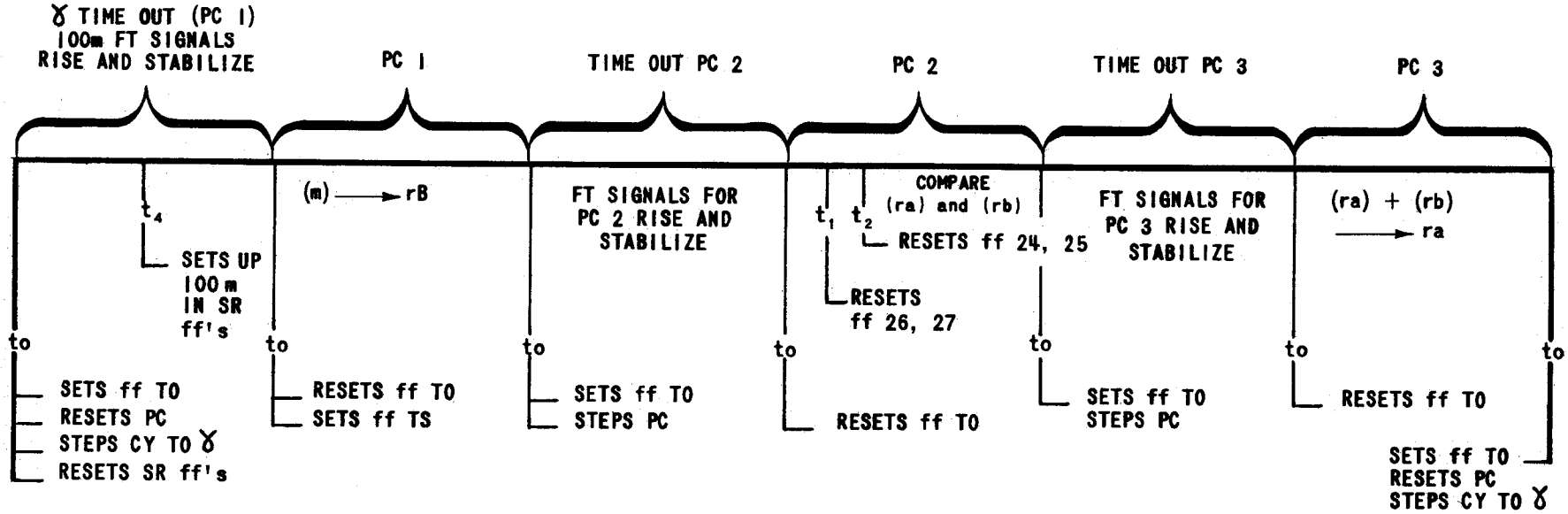
1) The contents of memory location M is transferred to register B.
2) The contents of rA and rB are sent to the comparator to determine if addition or subtraction is to be performed.
3) The contents of rA and rB are sent to the adder and the sum (or difference) is returned to rA.

Each step is executed sequentially which means the function table signals for step 1 must occur before those for step 2, etc. The program counter is a two-stage binary counter which keeps track of the current stage of the several multistage orders (add order and T order). It is shown on chart B just above the center of the function table.

Figure 37 is the timing chart for the add order. We start by assuming the ending pulse, t_0, of β time has stepped the cycle counter to 10, γ, set flip-flop time-out, reset the program counter, 00, and reset the static register flip-flops, the ending pulse is delayed four pulse times and emerges at time t_4 to sample gates 30A, 31A, ----, 36A in the static register circuitry. As noted in the β stage description, the p30 pulse position will be at gate 30A at the same time, t4, as the p31 is at gate 31A, ---, and the p36 is at gate 36A. Thus the delayed ending pulse "sets-up" the instruction 100 m in the static register flip-flops at time t_4 of the γ time-out minor cycle.

With the cycle counter reading 10 or γ, we obtain a signal on the γ line only which in turn excites the crystal tying it to the 100 or add line. With 100 m set up in the static register flip-flop 36 has a signal on its set line and flip-flops 35 and 34 will have signals on their reset lines. These static register lines thus excite their crystals tying them to the add line. The program counter was reset, reads 00, thus the left-most vertical line in the PC decoding function table carries a signal and excites the fifth crystal on the add line which then carries a signal to the encoding function table and thus allows function table signals 1, 3, 12, and 19 to rise to operating strength. As already mentioned, one minor cycle is allowed for this buildup--this is the purpose of the time-out signal, to prevent computer action till all the required F.T. signals are at full signal level. Before moving on to the next step, note that when the program counter reads 01 the 1, 3, 12, and 19 F.T. signals are dropped out and the 13, and 16 signals are picked-up. These signals disappear and we obtain the 10, 14, and 20 signals when the program counter reads 10. Thus the consecutive readings of the program counter will select the proper function table signals for each step of the add order.

The next t_0 will pass through gate 25 to reset FF TO, removing the time-out signal. Meanwhile the channel selection digits of m that were set up in flip-flops 32A and 33A have been decoded to select one of the four channels signal C_0, C_1, C_2, or C_3. The time selection digits set up in the static register flip-flops 30A and 31A are compared with that of the time selection counter (TSC-1 and 2) continuously. As already noted, the time selection counter is stepped every minor cycle by a signal occuring on the t_2 tap in the cycling unit. Thus during some minor cycle the time selection counter will be stepped into agreement with the readings of FF 30A and 31A. When this occurs none of



TIMING DIAGRAM FOR THE ADD ORDER 100 m
(SHOWN FOR NO LATENCY TIME ON PC 1)

FIGURE 37

the gates 70, 71, 72 or 73 will be open and thus there will be no inhibition on gate 1A from the left. The 1 function table signal being present will then allow the next t_o to pass through G1A and set flip-flop time selection (FFTS) producing the TS signal. The FT signal 3 allows gate 3 to pass the TS signal whence it is known as the TS_o (time selection signal for read-out).

Turning now to chart A, the minor cycle in which the TS signal is present is called the time selection minor cycle. TS and the channel selector signal open the proper G_t which passes a signal to open G_o with TS_o. Since TS is obtained by a t_o, the output gate of the selected memory channel is opened just at the proper instant of time so that the word* passes onto HSB1M p_o first, p_1, second, ---, p_{41} last. Now even if the output gate does not open at exactly t_o, say, t_4 the word appears on HSB1M in its correct order since p_o - p_6 consists of binary zeroes and the output of a closed gate is binary zeroes. This is the purpose of the SBW, to provide a safety factor in the operation of the gates and flip-flops. As the selected word passes onto HSB1M, the input gate to rB, G12A, has been opened by the 12 FT signal, and the word then enters the recirculation path of rB. The old word in rB is wiped out by the gate 12B passing the 12 signal to inhibit the clear gate 62.

The next t_o from CU is passed by FT 19 and TS through G19 (chart B) to step the program counter and set time out. The TO signal closes the input gate and opens the clear gate of rB to allow the new word to recirculate. This same t_o is gated by G1B to reset FF TS, removing the TS and TS_o signals.

With PC2 reading 1 and PC2 reading 0 we select FT signals 13 and 16, and drop out signals 1, 3, 12, and 19. On this step we are going to compare the words in rA and rB. This comparison is necessary, since words may be either positive or negative, we cannot be sure when giving the order to add two quantities that we are going to actually DO addition, if the signs differ we will do subtraction instead. Thus the comparator will produce a signal when the signs of rA and rB differ, indicating that subtraction is to be performed rather than addition. Now subtraction is formed by complementing the smaller quantity and adding to the larger. The scheme is necessary since direct subtraction is difficult to mechanize due to borrowing. This process is examined in greater detail in chapter III. Therefore, since the device that does the complementation is located on one input to the adder, we must make sure the smaller quantity is always switched on to this input line. The comparator will produce a signal when rA is smaller than rB.

As soon as TO is removed on PC2, a t_1 resets FF 26 and 27 and a t_2 resets FF 25 through gate 14B and resets FF 24 through G14A. The word in rA is applied to the half adder and G57B in the comparator at the same time that rB is also applied to the half adder and G57A. The reference timing is $p_o = t_{40}$. Thus the half adder will have an output during t_{35} if and only if the sign position, p37, of both words differ. This signal if present passes G55 to set FF25. If the signs do not differ FF 25 remains reset. Meanwhile, rA and rB have been feeding into the binary magnitude comparator (gates 57A and B and FF 27). The operation of this circuit was covered earlier. The single output line from FF 27 will carry a signal if $|rA| > |rB|$. Gate 56 samples the state of this flip-flop at t_{34}, after the p_{36} positions (most significant digit) has been

* Note that when TSC1 reads 0 and TSC2 reads 0, say, this means that word 00 of every memory channel is not now passing the output gates, but it means that the p_o pulse of these words will be at the output gate at the following t_o.

compared. If $|rA| > |rB|$ then G56 will be open at t_{34} to pass a pulse to G13A. This gate will be alerted by the 13FT signal to pass the delayed t_{34} to set FF 24. This FF will remain reset if $|rA| \leq |rB|$. Now the function of G65, FF26, and G16 is to assure us that the adder never produces a negative zero as a sum. The operation of this circuit is left as an exercise.

Again, the next t_0 is used to step the program counter and set TO. This time it is gated through G13B (chart B) since time selection is not required. The presence of time out prevents FF 24 and 25 from being reset, and wiping out the results of our comparison.

PC3 produces FT signals 10, 14, and 20. The 14 signal again prevents FF 24 and 25 from being reset when time out is removed. FT signal 20 alerts gates 20B and 14. If FF 25 was set, indicating the signs of rA and rB differed, gate 20B passes this signal whence it is called the CP or complementer signal.

Likewise if FF 24 is set it meant that $|rA| > |rB|$ and therefore we will switch rB onto the lower input to the adder. In this case gate 14 does not have a reset signal to pass. That is, there is no S_2.

As soon as time out is removed on PC3, gates 20C and D open and let the contents for rA and rB into the adder circuits. The contents of rA pass through gate 58A onto the upper input path of the binary adder if the S_2 signal is not present. Otherwise, it enters through gate 59B and thus must pass through the half adder and G54 to enter BA on the lower path. The contents of rB are applied to gates 58B and 59A. The contents of rB take the upper path into the adder if S_2 is present, and the lower path through the complementer when S_2 is absent. The action of the complementer was described earlier. When the CP signal is present G53 passes the t_6 through t_{35} pulses from the cycling unit to form a second input to the half adder whose output then is the binary complement minus 1 of the quantity coming through gates 59A or B.

If the CP is not present the output of these gates enters the binary adder unchanged. In any case gate 54 strips out the sign pulse of this BA input. As the quantities enter BA their sum is returned to rA through gate 20E, the old word in rA being prevented from recirculating by the inhibition on the clear gate, G61, from G4B. By checking the timing it can be seen that the sum returns to rA in synchronism with the old word being wiped out. Binary carry from the adder is fed back as a third input after being delayed one pulse time (one binary column). If subtraction is being done, the CP signal is present. This allows gate 52 to pass a pulse onto the carry line converting the binary complement minus 1 to the full binary complement. Since the sign of the lower input to BA is deleted by G54, the sign of the sum will always be that of the upper input. This corresponds to the rules for algebraic addition. Note that if addition is actually being performed, a carry from the most significant binary column, p36, indicates an overflow (i, e, an exceed capacity). This carry is allowed through G50 to set the overflow flip-flop. The O. F. signal prevents G25 from passing a t_0 to reset T.O. after it has once been set, thus effectively stopping the machine. Of course, if subtraction is being performed, an overflow is expected, thus the CP signal prevents setting the overflow flip-flop, FF28. FT signal 10 gates the next t_0 as an ending pulse which sets the time out FF, resets PC, and passes through G41 to step CY to 00 or α.

The remaining seven orders are of less complexity and are left to the reader as an exercise. It should be remembered that the purpose of this chapter is to present fundamental concepts and terminology used in the design of UNIVAC. If an understanding of the high points of the material has been gained; the purpose of timeout the cycle counter, function tables, etc; the reader may ignore, or rather forget, the details.

CHAPTER III

INTRODUCTION TO UNIVAC LOGIC

SECTION I

Representation of Information

The binary method of representing numbers was introduced in Chapter II. This method is almost universally used in large computing machines. The simple computer described in that chapter is called a binary machine because the various ones and zeros making up its "word" had a place value that represented some power of two. For example, in the binary number 1011 the left-most "one" has the value 2^3 or 8.

Another number representation using ones and zeros is also possible. This is called the "coded decimal" notation. Any number between 0 and 9 may be represented by at most four binary columns as the following table shows.

Decimal Digit	Binary Equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Now consider the decimal number 147. If this number were expressed in binary notation it would appear as 10010011. However, we could represent each of the individual digits in binary fashion and yet retain their decimal value as regards position. Thus:

0001 0100 0111

1 4 7

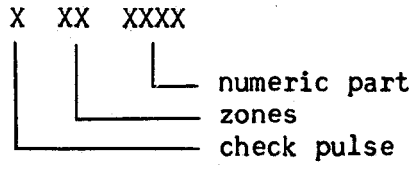
This coded decimal notation is more convenient to use than pure binary, mainly because humans can read the numbers easily. With only a very little practice one can recognize the coded decimal representation 0010 1000 0111 0110 as being 2876, but the same amount of practice does not yield equal results for its pure binary form 101100111100.

A simple modification of the coded decimal notation is used in the UNIVAC. It is called "excess-three" notation. If the number X is to be represented in the computer, its binary representation is that of the number X+3. The excess-three notation for the decimal digits is shown in the following table.

Decimal Digit	XS-3 Equivalent
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

This notation permits considerable simplification in the arithmetic units of the UNIVAC.

Since UNIVAC was intended as a universal computing device two additional binary columns called zone indicators are included in order to represent letters of the alphabet. And a further, third, binary column called the check pulse is included for checking purposes. Thus a complete UNIVAC character or digit consists of seven binary columns:



The check pulse is a binary one or zero as required to make the sum of the binary ones in each digit odd. As an example:

7 = 1 00 1010
B = 0 01 0101
Z = 1 11 1100

Chart C in the Appendix depicts the numeric part and zone indicators for all 63 characters represented in the UNIVAC. The check pulse is not shown but can be determined by the rule given above. Note that all numbers have 00 zones.

SECTION 2

Mode of Operation of the UNIVAC

A complete discussion of the operation of the UNIVAC requires a detailed description of the four-stage cycle of operations and each of the forty-three instructions in a manner similar to that contained in Chapter II. This is quite beyond the scope of this book, but a brief description will suffice for operating personnel.

Before covering an analysis of the mode of operation we shall briefly review the word composition and organization of the memory. A UNIVAC word is the fundamental unit of memory and consists of twelve characters or digits. A digit is any one of the 63 listed in Chart C. The digits in a word are numbered from left to right:

X	X	X	X	X	X	X	X	X	X	X	X
1	2	3	4	5	6	7	8	9	10	11	12

When a word consists of numbers, position 1 contains the sign (zero for plus, - for minus) and the decimal point is considered to be just to the right of the sign. Thus UNIVAC treats all numbers as being less than unity. As mentioned in Section 1, each character is composed of seven binary bits.

The main memory of the UNIVAC is set of 100 acoustic delay lines called channels. Each channel has a capacity of 10 words. In addition, there is a ten word channel, register Y, a two-word register V, six one-word registers A, X, L, F, CC, and CR, and a half-word register SR. (Chart D in the Appendix) In the input circuitry there are six ten-word channels known as register I; and similarly there are six ten-word channels, register O, in the output section of the computer. The essential components of each channel and register are identical with the acoustic memory of the small computer described in the preceding chapter, as shown in Figure 1.

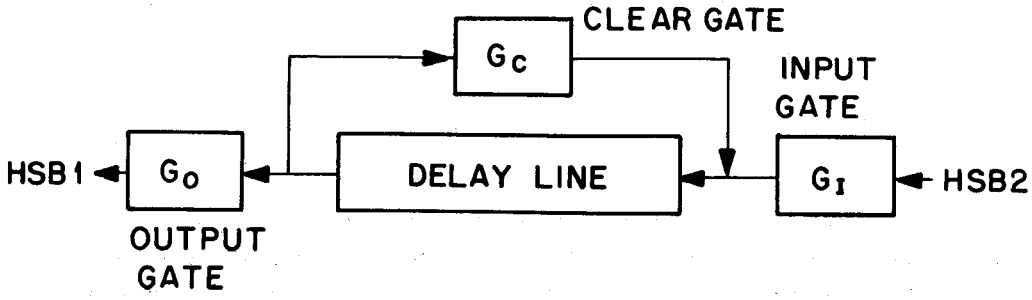


Fig. 1

The time necessary for the voltage train representing a complete word to pass a given point in the computer is known as a minor cycle, and is a fixed interval of time. The cycling unit (CU) emits signals marking the elapse of each minor cycle and generates the timing signals needed for the proper operation of all units.

The word positions in each ten word channel are numbered from zero to nine. The Cycling Unit assures the synchronization of all ten word channels, so that when word 5 of channel 00 is ready to emerge from its delay line, word 5 of each and every 10 word channel is ready to emerge from its respective delay line. Further, when digit position 12 is ready to emerge from register A, digit position 12 of some word in every delay line is ready to emerge. Words travel through the computer least significant digit first, that is digit position 12, first, followed by 11, 10, --- 1. Also, within each decimal digit the numeric part preceeds the zones which precede the check pulse.

Since the only time a word may be brought out of the memory is when it is passing by the output gate some means must exist for determining when the desired word may be operated upon. In order to identify word 073 it is not enough to specify that the word is in channel 07. We must know at what time the third word in the channel is ready to emerge. The Time Selection Counter (TSC) counts the number of minor cycles elapsed modulo ten. Thus when TSC reads 3 the third word of each ten word channel is ready to emerge from its delay line. The Time Selection Counter and the devices for selecting a particular channel of the main memory are called the memory switch.

The normal operation of the computer involves four distinct stages called α, β, γ, δ, and are executed in that order. The cycle counter CY is a two stage binary counter (four stable states 00=α, 01=β, 10=γ, 11=δ) which indicates the current stage the computer is on. Each stage is further divided by "time-out" periods, TO, as follows:

CY Readings	Stages
	α TO
00	α
	β TO
01	β
	γ TO
10	γ
	δ TO
11	δ

The purpose of these several divisions will become clear as the characteristics of each stage are considered. Also, in the discussion to follow, it will be helpful to the reader to refer to the 'Simplified Block Diagram of UNIVAC' - Chart D in the Appendix.

To begin, assume that CY is in the α state and the time out flip-flop is set, producing the α TO period. When CY reads α(00) it causes the function table to generate certain signals associated with the α state. The presence of the time out signal which lasts for one minor cycle prevents any further action of the computer and is necessary to allow the function table signals to rise to their normal voltage levels. Thus one minor cycle elapses and then a pulse from the cycling unit removes the time-out signal by resetting flip-flop TO. Removal of TO allows the α function table signals to open the output gate of the Control Counter (CC) whence the word in CC passes out onto High Speed Bus IA, (HSB1A)

into the High Speed Bus Amplifier (HSBA) which under the action of the α function table signals switches the word onto HSB2A. While the word is on HSB2A it is examined by the HSB Odd-Even Checker (HSB O-E C). As each character enters HSB2A the number of binary ones is counted by O-E C. If for any digit an even count is registered, an error signal is produced stopping the computer on the next time out period. The word from CC enters HSB2A and then the Control Register CR, whose input gate has been opened by an α FT signal. It passes through CR and enters the Static Register SR. SR consists of a set of flip-flops and at the proper time a pulse from CU transfers the characters in 7 through 12 from their dynamic form as a voltage train into static form in the SR flip-flops. At the same time a pulse from CU is allowed to step CY to β (01) and set FFTO. The word in CC will appear as:

000000000XXX

where XXX is a number between 000 and 999 and thus 000XXX is set up in SR at the end of α time.

When CY was stepped to β (01) the β function table signals are generated. Again the presence of TO prevents any action of the computer for 1 minor cycle in order that the FT signals have time to rise to full strength. During each minor cycle the reading of TSC and the right hand digit (time selection digit) in the Static Register are compared. During the minor cycle in which they agree (TSC is stepped once each minor cycle, reading successively 0,1,2,---9), a flip-flop (FFTS) is set and in conjunction with the β FT signals a signal is applied to the output gate of each of the 100 channels of the main memory. Meanwhile the fourth and fifth instruction digits in SR have been decoded in the channel selection part of the memory switch and a signal is applied to the output gate of one particular channel; thus, only one channel during one minor cycle will have both signals present to open its output gate. This minor cycle is called the time selection minor cycle and may be as much as the 10th minor cycle after CY was stepped to β. The TS and β FT signals close the clear gate and open the input gate of CR. The word from the selected channel passes out onto HSB1M into the HSB amplifier where it is routed onto HSB2A. It is also examined by HSB O-E C. From HSB2A the word enters CR. During this TS minor cycle the output gate and a special input gate of CC are opened while the clear gate is closed. The contents of CC are routed to the adder by a special path where 000 000 000 001, generated by CU, is added to it and the sum coming from the adder returns to CC. At the termination of the TS minor cycle CY is stepped to γ and TO set. The presence of TO closes the output gate of CC, the input gate of CR, and opens the clear gates of CC and CR. The input gate of CC is closed somewhat later in order that all of the sum returning from the adder will enter CC.

Returning to the β TS minor cycle, the word from the memory enters the recirculation path of CR and it is also routed into the SR. At the same time, the ending pulse from CU which sets TO and steps CY will also set up the digit positions 1 - 6 in the SR flip-flops.

Recapitulating:

α TO: α function table signals rise to strength.
α : Digit positions 7 - 12 of CC set up in SR. Ending pulse from CU steps CY to β and sets TO.

βTO: β function table signals rise to strength.
β During β TO and each following minor cycle digit position 12 (TS digit) in SR is compared with TSC. Upon agreement the word in the memory location specified by digits 10 - 12 of CC (which was set up in SR during α) is transferred to CR and the left 6 digits (left instruction) transferred to SR. The contents of CC are augmented by 1. The ending pulse fromCU steps CY to γ and sets TO.

Attention is called to the similarity of stages α and β in UNIVAC to that of the elementary computer. However, the small computer has no device similar to CR, the instruction word coming from the memory entering SR directly. Since a UNIVAC instruction is completely defined in six digits, it permits two instructions per word. In order to speed computation, the number of memory "look-ups" should be reduced to a minimum. Thus by use of CR, it is possible to extract two instructions from the memory at one time. The instruction pair is stored in CR during β time, and the left hand instruction (digits 1 - 6) are sent to SR in time for the ending pulse that steps CY to γ and sets FFTO to set-up the instruction in the SR flip-flops.

During γ TO, the instruction in the SR actuates the function table directly to produce the signals peculiar to that instruction. After TO is removed the instruction is executed. CY, being on γ, produces a special function table signal that passes the γ ending pulse to set-up the right-hand instruction in SR at the same time as CY is stepped to δ and TO set. This same process is then repeated for δ time. However, the ending pulse now steps CY to α and the entire four steps are repeated.

SECTION 3

Analysis of Central Computer Instructions

The UNIVAC operator must know what each UNIVAC instruction does in order to perform his duties efficiently. Chapter II has described in detail the manner in which a serial computer similar to UNIVAC operates, so here we need merely list the UNIVAC instructions and give a brief account of each. Each instruction will be broken down into its separate Program Counter steps.

As noted in Chapter I, each instruction is designated by a first instruction digit, a second instruction digit if needed, and a fourth, fifth, and sixth instruction digit which generally specify a memory address and are collectively indicated as m. The contents of memory location m are indicated by (m), and the contents of a special register, such as rA, by (rA). The arrow, $\longrightarrow$, indicates that the word on its left replaces the contents of the memory location or register on its right.

The UNIVAC programming manual contains examples of the use of each instruction as well as additional notes on programming. The instructions described here are called central computer instructions, the remaining instructions for input-output are discussed in Chapter IV.

TRANSFER INSTRUCTIONS:

A) From Memory to Registers

Instruction and Name	Program Counter Steps: Step	Operation
B m "Bring"	1	(m) → rA, rX
L m	1	(m) → rL, rX
F m	1	(m) → rF
V m	1	(m) → rV } Two word transfer. See programming manual for restrictions on m.
	2	(m+1) → rV
Y m	1	(m) → rY } Ten word transfer. See programming manual for restrictions on m.
	2	(m+1) → rY
	•	•
	•	•
	•	•
	10	(m+9) → rY

B) From Registers to Memory

Instruction and Name	Program Counter Steps: Step	Operation
C m "Clear"	1	(rA) → m zero → rA
H m "Hold"	1	(rA) → m
G m	1	(rF) → m
J m	1	(rX) → m
W m	1	(rV) → m } Two word transfer. See programming manual for restrictions on m.
	2	(rV) → m+1
z m	1	(rY) → m } Ten word transfer. See programming manual for restrictions on m.
	2	(rY) → m+1
	•	•
	•	•
	•	•
	10	(rY) → m+9

C) From Registers to Registers

Instruction and Name	Program Counter Steps Step	Operation
K m	1	(rA) → rL zero → rA

D) Special Transfers

Instruction and Name	Step	Program Counter Steps Operation
R m	1	(CC) → m. During the transfer A U is placed in the seventh digital position.
E m "Extract"	1	The least significant binary bit of each digit in rF is examined. If it is a zero, replace the digit in the corresponding position of rA by the digit in m.
.n m	1	Shift (rA) one digit position to the right replacing position 1, which is now empty, with a zero.
	2-n	Same as PC 1
;n m	1	Shift (rA) one digit position to the left replacing position 12, which is now empty, with a zero.
	2-n	Same as PC 1
-n m	1	Shift all positions, except sign, of (rA) right one digit position. Place zero in position 2 which was empty.
	2-n	Same as PC 1
On m	1	Shift all positions, except sign, of (rA) left one digit position. Place zero in position 12 which was empty.
	2-n	Same as PC 1

ARITHMETIC INSTRUCTIONS:

Instruction and Name	Program Counter Steps Step	Operation
A m "Add"	1	(m) → rX
	2	(rA) + (rX) → rA
S m "Subtract"	1	-(m) → rX (rA) + (rX) → rA
X m	1	(rA) + (rX) → rA
M m "Multiply"	1	(m) → rX
	2-4	3|rL| → rF 050 000 000 000 → rA (Round-off)
	5-15	Compute (rL) x (rX) by repeated additions of (rL) and (rF) to rA. Product is in rA.
N m "Negative Multiply	1	-(m) → rX
	2-15	Same as M m.
P m	1-15	Same as M m except in PC-4 the round-off quantity is not placed in rA. The multiplication process thus develops a 22 digit product. The most significant 11 digits and sign are placed in rA, the least significant 11 with sign in rX.
D m "Divide"	1	(m) → rA
	2	Determine sign of quotient from (rA) and (rL)
	3-14	Compute $\left\lvert\frac{(rA)}{(rL)}\right\rvert$ by repeated additions and subtractions and place in rX.
	15	(rX + 000 000 000 005)(round-off) → rA
	16	Insert sign of quotient in sign position of rA and rX

CHOICE INSTRUCTIONS:

Instruction and Name	Program Counter Steps Step	Operation
00 m "Skip"	1	No operations performed
U m "Unconditional Transfer of Control"	1	Digits 10, 11, 12 of (CR) Digits → 10, 11, 12 of (CC)
Qn m	1	Compare (rA) and (rL). Stop computer if breakpoint button n is depressed.
	2	If PC 1 indicated (rA) = (rL) execute U m instruction, otherwise do skip instruction.
Tn m	1	Same as PC 1 of Qn m instruction.
	2	If PC 1 indicated (rA) > (rL) execute U m instruction, otherwise do skip instruction.
9 m "Stop"	1	Stop computer by setting stop flip-flop and time-out.
, m "Comma	1	If comma breakpoint switch is set stop computer, otherwise treat as a skip.

OVERFLOW: Overflow is caused by a carry from digit position 11. This may occur on the A, S, X, or D instructions. In the event of an overflow the normal operation of the computer on the next α and β time is blocked:

1) The transfer of (CC) to SR during the next α stage is blocked. Instead zeros are placed in SR.

2) The addition of one to (CC) is blocked during the next β.

3) Step 1) then, makes the computer execute the instructions in memory cell 000.

4) After completing the instruction pair in 000 the computer does normal γ and δ stages. The next instruction pair being determined by (CC). If however, the second instruction digit of an A, S, X, or D instruction is a - (minus sign), the computer is stopped on the next time out.

CHAPTER IV
INPUT-OUTPUT ORDERS

SECTION 1

Introduction

The previous chapter has discussed the internal operation of the UNIVAC. In this chapter we shall be concerned with the manner in which the operating and maintenance personnel obtain contact with the UNIVAC. Since we cannot "read" the information stored in the memory directly (it is a voltage waveform), peripheral equipment is provided for producing printed information, often called "hard copy" from the computer, and for inserting information into the computer (data and instructions).

For printing very small amounts of output data an electric typewriter is connected directly to the central computer, while a similarly connected keyboard is provided for small input amounts. The main communication chain, however, is through the media of magnetic tape. No attempt is made to provide a description of the terminal equipment for magnetic tape in this manual. The equipment consists of:

For input---Unitypers which transcribe printed source documents onto tape and Card-To-Tape Converters for transcribing punched cards.

For output---Uniprinters and High-Speed-Printers for transcribing tape into printed copy, and Tape-To-Card Converters for transcription of tape to punched cards.

SECTION 2

Tape Recording

Before outlining the input-output orders we must consider the manner in which information is represented on magnetic tape. Just as binary information is represented internally in a computer by a voltage, binary information can also be represented by the different orientations of a small magnet on a metallic strip. Physically, a UNIVAC tape is a thin strip of phosphor-bronze about .002 inches thick, $\frac{1}{2}$ inch wide, and as much as 1500 feet long. This tape is coated with a special metallic film which can be magnetized. Each decimal digit is recorded across the width of tape. Thus a word such as 012 345 ACZ. 9 might appear as:

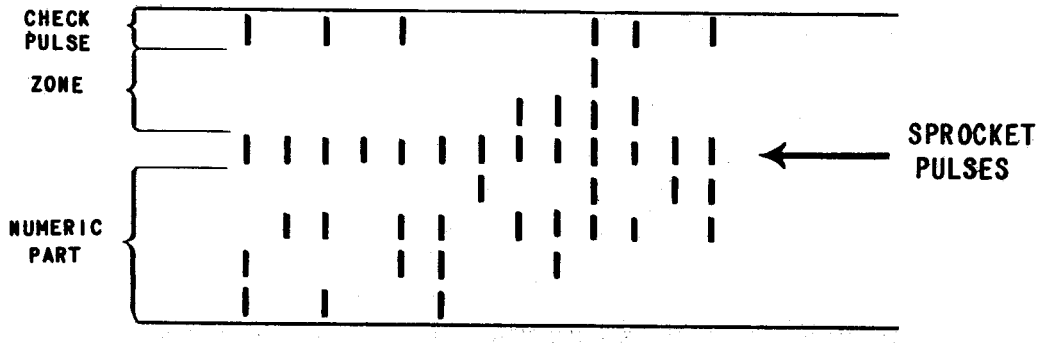


FIG. 1

As noted, an additional channel of pulses is recorded for each digit. These pulses, known as sprocket channel pulses, indicate that a digit is present in this column. Each spot is actually a small magnet impressed on the metallic film coating of the tape. For each word, the first digit recorded is the sign digit (digit 1), followed by digit 2, and finally the least significant digit (No. 12). Recording can be done at a density of 128 digits per inch*. Now in

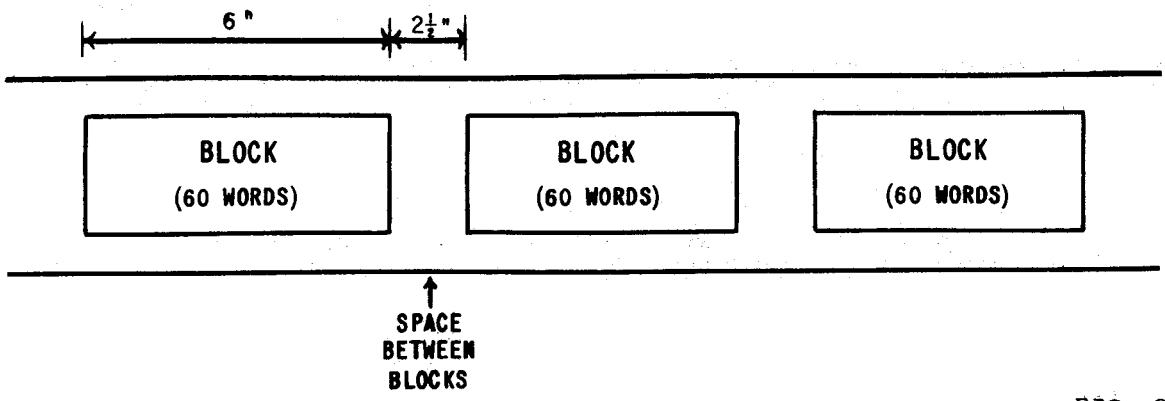


FIG. 2

*There is also a recording density of 20/inch used for Uniprinting.

order to achieve rapid reading and writing rates, UNIVAC records and reads information in blocks of 720 digits (60 words). When recorded at a density of 128 digits/inch, a block occupies about 6 inches of tape. In addition there are 2.5 inches of blank tape between each block to allow for the starting and stopping space required by the Uniservo (The tape read-write mechanism).

Tapes may be supplied in several lengths; the longest can record as many as 2000 blocks or 1,440,000 decimal digits.

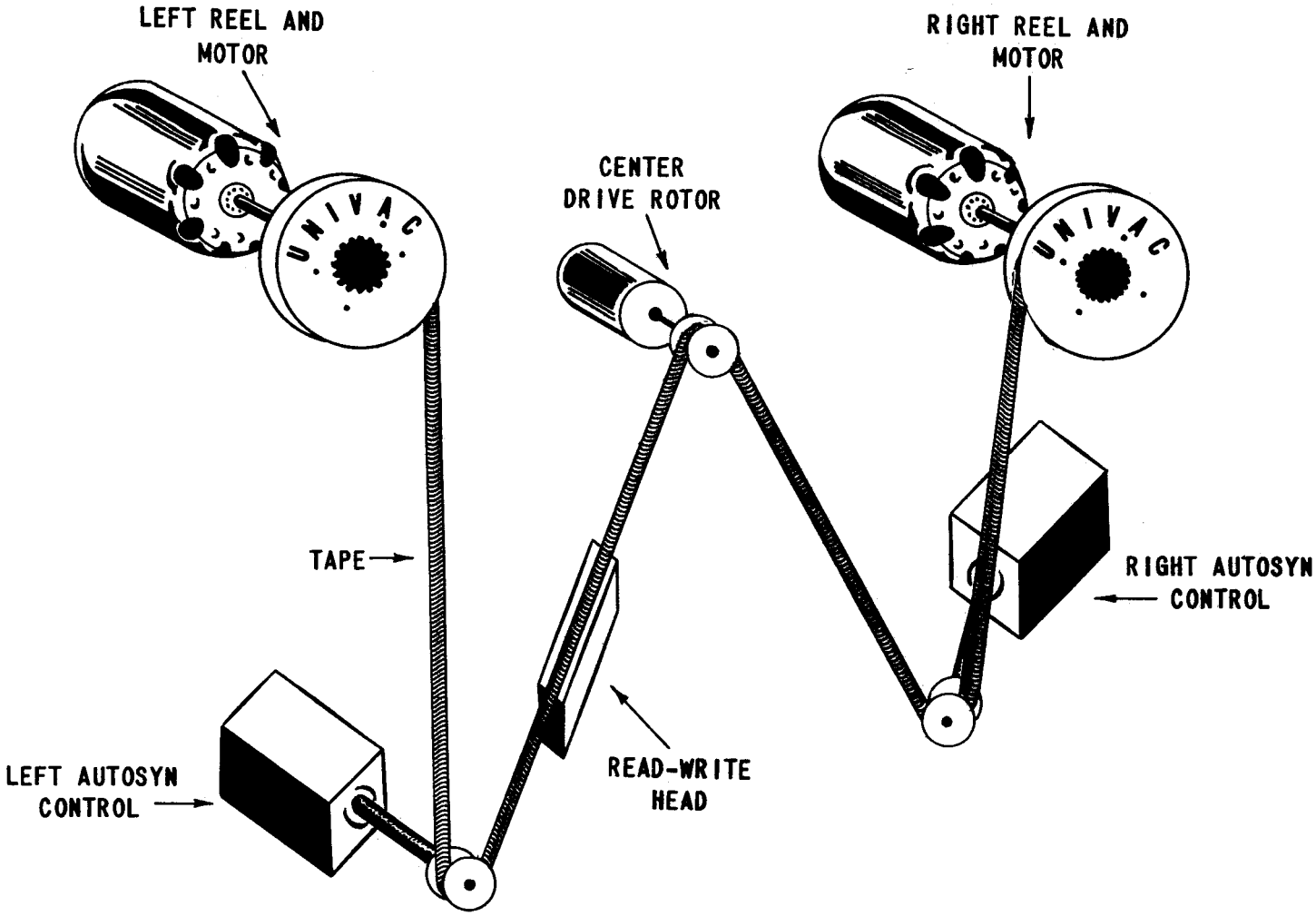
SECTION 3

Uniservo

The Uniservo is the device linking the central computer (the memory, arithmetic and control units) to the magnetic tape. The Uniservo consists of a tape-handling mechanism and a read-write head consisting of eight separate read-write coils-one for each binary channel on the tape. When we wish to read information on a tape, for instance, we move the tape rapidly under the reading head. The presence of a magnetic spot on the tape causes a current to flow momentarily in the reading head coils. This current is then translated into the binary voltage representation used inside the computer. This process will be covered in more detail in the next section.

To provide for a reliable signal to the read-write head and to reduce the time necessary to read or write a block, the tape must be rapidly accelerated to its running speed of about 100 inches/sec, and just as rapidly decelerated to a stop. The tape is wound on a reel and may weigh up to 3 pounds. If the tape were unwound directly from the supply reel, passed across the read-write head, and wound on the take-up reel, the time to accelerate to full speed and then stop would be undesirably long, since the motors moving the heavy reels are, necessarily, large. For this reason a method has been developed which permits the rapid acceleration of a short section of tape, thus allowing sufficient time for the heavy reels to reach full speed.

Figure 3 is a simplified picture of the tape handling mechanism. Three motors, left reel, center drive, and right reel, are provided for moving the tape. The two pulleys at the bottom of the left and right loops of tape are floating, that is, they are not fastened rigidly to the Uniservo panel, but may move up or down. When the computer orders a block of information to be read from or written on tape, a signal is sent to start the center drive motor. This small motor rapidly reaches full speed, drawing tape from the large left hand loop across the read-write head (where it is read or written upon), and dumping it into the short right hand loop. Since the pulleys are free to move, the center drive has very little drag placed on it by the tape, the loop system actually provides slack in the tape. As tape is drawn out of the left loop, the loop shortens, pulling up the pulley. An arm is attached to the pulley and to a device called the left autosyn. As the pulley begins to rise, because of the shortening loop, the arm rises. This change of position of the arm detected by the autosyn, which causes the left reel motor to start paying out more tape. The reverse action, meanwhile, occurs in the right-hand loop. As the center drive spills tape into this loop, it grows larger, lowering its pulley. This pulley drops its attached arm, causing the right autosyn to start the right reel motor, thus taking up the extra tape. The left loop is allowed to become



SIMPLIFIED DIAGRAM OF UNISERVO

FIG. 3

short and the right reel long as soon as the reel motors reach full speed, thereby placing them in the proper position for stopping. When one block has been read or written, the computer sends a signal stopping the center drive motor. Since the large reel motors do not decelerate as rapidly as the center drive motor, the left reel motor will pay out tape, filling out its short loop, while the right reel motor will take up tape, shortening its long loop.

This discussion applies to tape motion from left to right. When we wish to move the tape backward, from right to left, the computer causes the reel motors to make the left loop short and the right loop long, and the center drive is run in the reverse direction. Direction reversal takes .6 seconds.

A reel of tape is always mounted on the left reel motor. The end of the tape is hooked to a leader which is already threaded through the pulley system. The servo is then said to be in "First Block Condition". When the center drive starts moving the tape, the reading and writing of information for the first time only is delayed for about 1½ seconds, until the leader has passed the head.

SECTION 4

Input-Output Control

A sizable portion of the UNIVAC's equipment is concerned with reducing the time necessary to record or receive information from magnetic tape. This is the purpose of the complicated Uniservo mechanism and the reading and recording of information in blocks.

Special input-output control circuitry is also provided for this purpose. This circuitry allows the computer to continue with other internal computations while a block is being read from tape or is being recorded.

When a tape write order is given, a block (60 words) is transferred from the 1000 word memory to the block-length output register, rO, and a signal is sent to start the desired Uniservo. The central computer is then released to continue executing programmed instructions while the output control circuits and the selected Uniservo are engaged in writing the block in rO on tape. The write operation can be done in the forward direction only. After the center drive has accelerated the tape to full speed, the output control circuits peel off the 1st digit (sign digit) of the first word of rO, and set it up in output flip-flops, which then cause current to flow in the coils of the read-write head. This current induces magnetic spots on that part of the moving tape directly under the r-w head. Then the second digit of the first word is set up in the flip-flops and written on tape. This process continues until the last digit (digit No. 12) of the 1st word is written. The process is repeated for the second and following words. When the last digit of the 60th word in rO has been written, the output control circuits emit a signal stopping the center drive motor (and therefore the tape).

The reverse operation occurs in reading a block already recorded on tape. A signal from the central computer starts the tape moving, after which the central computer moves on to the next instruction. As the tape passes the r-w head, the decimal digit on the tape induces a current in the head coils, which set up the binary bits in input flip-flops. The digit is then placed in the first

digit position of a special one word register* and the next incoming digit is set up in the flip-flops. Each digit is thus read from tape and assembled into a word. When the 12 digits of the word are assembled, the word is placed in the 60 word input register, rI. This process continues until all sixty words have been read into rI. When the last word of the block has been assembled, a signal is emitted which stops the tape. The block on the tape has now been duplicated in rI. When reading a block from tape which is moving in the backward direction, the first digit picked up is of course the last digit of the 60th word. The input controls in this case place the digit in the last position of the special one word register; and, when the word is completely assembled, place it in the 60th position of rI, etc. Thus, although the block is read in the reverse order of digits the information appears in the input register in normal order, just as though the block had been read forward.

SECTION 5

Interlock

In order to speed the overall operation of UNIVAC, the central computer was released very early in the write (read) operation, so that it might continue computing. To prevent the computer from attempting to make use of a block not yet completely read from tape or to begin another write (read) order before r0 has been emptied, (or rI filled) an elaborate interlock system has been designed.

Before the 60 word transfer from memory to r0 takes place in a write order the following interlock tests are performed:

1. Is a write order still in progress?
2. Was the last write order improperly executed?
3. Is the selected servo already engaged in a write, read, or rewind order?
4. Is there a ring on the reel mounted on the selected servo to prevent writing?

If any of these conditions exist (i.e. answered "yes") the computer is not released and thus "stalls" on the write order until the conditions are removed.

In the read order the interlock tests performed are:

1. Is a read order still in progress?
2. Was the last read improperly executed?
3. Is the selected servo already engaged in a read, write, or rewind order?

If any test is answered affirmatively the computer stalls on the current read order. By examining the interlock test we see that a read operation can be ordered, and while it is in progress we can order a write operation (on different servos) without stalling the computer until the read is completed.

By means of the special devices described, UNIVAC has an excellent match between its internal and external operations. The time required to read or write a block is approximately 100 milliseconds.

*The process is actually a bit more complicated

SECTION 6

Input-Output Orders

The number of Uniservos included in a UNIVAC system is variable. As many as ten can be connected to the central computer. They may be used to read or write in arbitrary order, so that what a particular servo is doing, reading or writing, at any instant depends on the program being executed. Also, no central computer processing may be done on information when it is in r0 or rI. Again r0 is cleared to binary zeros at the termination of every write order, while rI is cleared to binary zeros after every transfer of rI to the memory. In the following orders, n, the second instruction digit designates any one of the ten Uniservos 1, 2, ...9 - (The tenth is designated by the minus sign). The

1m 000	Fill rI by reading 1 block forward on Uniservo n.
2n 000	Fill rI by reading 1 block backward on Uniservo n.
30 m	Transfer rI (60 words) to the memory, placing the first word of rI in cell m. The second in cell m + 1, -- , the 60th in m + 59.
40 m	Exactly identical to 30 m instruction.
3n m	Execute the instruction 30 m, then upon completing the transfer, execute a 1n 000 order.
4n m	Execute the instruction 30 m, then upon completing the transfer execute a 2n 000 instruction.
5n m	Transfer the 60 words in memory cells m, m + 1, ---, m + 59 to r0. Then write the contents of r0 on Uniservo n at a density of 128 pulses/inch.
7n m	Same as the 5n m instruction except the writing density is 20 pulses/inch. The computer will read at either density, but the Uniprinter only at the low density. This order is used when information is to be Uniprinted.
6n m	Rewind the tape on Uniservo n. This places Uniservo n in "first block condition."
8n m	Same as 6n m instruction but places an interlock on servo n that prevents the computer from reading or writing on this tape. The interlock is released when the operator removes the tape.

SECTION 7

General Information

When a reel of tape is coated, its recording characteristics are tested. For certain reasons some areas of the tape may not reliably hold a signal -- the noise level may be too high, or a splice may be present. Holes are punched in the tape throughout those areas. Photo-electric cells, mounted to the left and right of the Uniservo read-write head to sense the presence of such areas, automatically interrupt the reading or writing until the area is passed over.

As previously mentioned, mounting a new reel of tape merely requires that the operator latch the reel onto the left reel motor and hook the end of the tape to the already threaded leader. After very little practice, an operator can change tapes in a half minute. In order to prevent physical damage to the tape, the leader and the inside end of the tape (the end attached to the reel) have rubber lugs attached. If a faulty program tries to read or write off either end of the tape the lugs trip a micro-switch which cuts off power to the servo.

Figure 4 is a front view of a Uniservo. There are seven lights on the servo which indicate its current state. When the servo is energized, servo power on, the green ready light will be lit (#1 - Fig. 4). If no other lights are on this signifies that the Uniservo is in first block condition, which means the tape is in the rewound state ready to read or write the first block. When the computer is executing a magnetic tape instruction the following additional lights will be lit:

If Uniservo is engaged in:

Read forward - lights 6 and 4
Read backward - lights 6 and 5

Write - lights 7 and 4

Rewind no interlock - light 2
Rewind interlock lights 2 and 3 (light 3 remains lit until tape is removed)

Chapter V discusses the error circuitry in detail, but it is convenient to note here that if an error is made by a Uniservo the appropriate lights will remain lit and this enables the operator to determine at a glance what Uniservo made the fault and what operation was being performed.

As far as the UNIVAC is concerned, Uniservo 1 need not be the first Uniservo in the line. A plug board is provided in the computer to enable a person to interchange any two servos without physically moving them. This is a great convenience in averaging the daily use of each servo, or for replacing a servo temporarily with a spare while it is undergoing repair or adjustment.

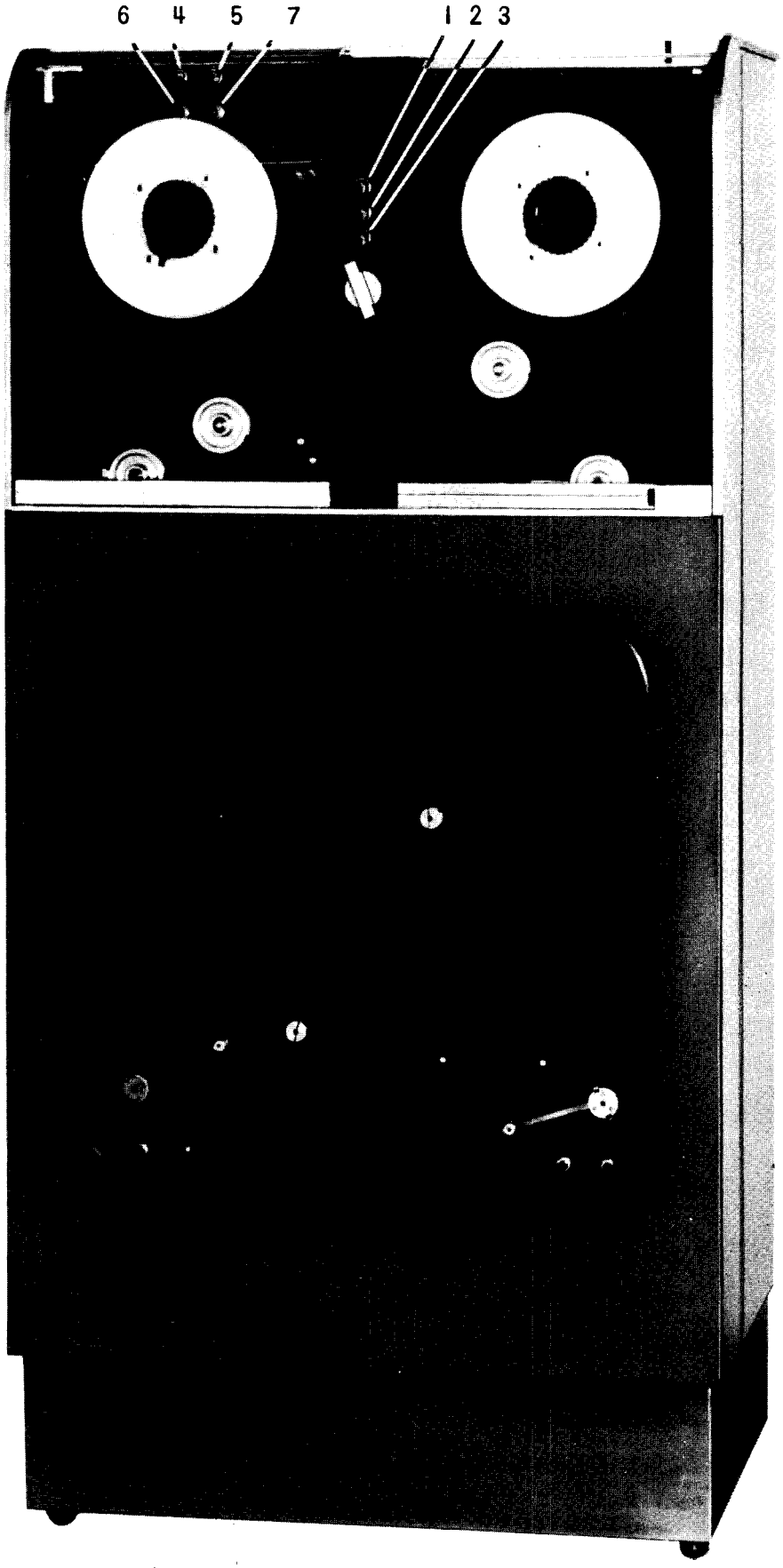


FIG. 4

SECTION 8

Keyboard Input, Typewriter Output

Magnetic tape forms the prime method of getting information into or out of the computer. A keyboard (see chart in appendix) on the Supervisory Control Console provides direct input for engineer maintenance or program requirements.

10 m	Type one word from Supervisory Control Keyboard into a special input register; When the "Word Release Key" is struck the computer transfers the word into memory cell m and proceeds to the next instruction.

An electric typewriter attached to the Supervisory Control is provided for direct visual output. When a word is being typed in from Supervisory Control (10 m order) it is also printed on the S.C.P. for a visual check. Print-outs may also be made by using the:

50 m	Print on the Supervisory Control Printer the word in memory cell m.

In discussing the print order, it is necessary to examine the action of the printer when it receives certain UNIVAC characters. This discussion applies to the Uniprinter as well. This is a separate device for transcribing information recorded on magnetic tape at a density of 20 pulses/inch into printed copy.

All of the UNIVAC characters received by a printer are printed except for the following:

SYMBOL	NAME	PRINTER'S ACTION (Normal)
✗	Ignore	Causes printer to ignore it and pass on to the next character. Nothing is printed nor is the carriage moved.
Δ	Space	Action is the same as depressing space bar on typewriter. Nothing is printed.
✗	Carriage Return	Causes carriage to be returned to the left margin stop and gives a line feed (the number of lines fed depends on the manual setting of the line-feed lever for either 1, 2, or 3 lines).
✗	Tab	Action is the same as depressing the tab key. Tab settings must be made manually.
✗	Shift Lock	Action is the same as depressing shift lock key. Prints upper case of all characters following it.
✗	Un-Shift	Releases the carriage from printing upper case.
✗	Printer Stop	Causes printer to stop printing.

⁄B	Printer Breakpoint	Is interpreted as a ⁄P if breakpoint switch on printer is set to breakpoint. If switch is on normal it is treated as an ⁄I.
⁄Ø	Single-Shift	Prints only the next character in upper case.

Some cases arise (when proof reading for instance) where it is desirable for the above characters to be printed and not to perform their normal functions. By rotating a switch on the printer to "Computer Digit" each of the above characters print the following symbols:

⁄I	Prints x
△	Leaves space as on Normal
⁄R	Prints /
⁄Y	Prints *
⁄L	Prints z
⁄Ø	Prints 8
⁄P	Stops Printer as on Normal
⁄B	Prints y
⁄Ø	Prints -

CHAPTER V

SUPERVISORY CONTROL
AND CHECKING CIRCUITS

SECTION 1

Introduction

In Chapter II a simplified computer was described in considerable detail. Study of that chapter permitted the development of fundamental electronic elements and concepts such as the Static Register, the Cycle Counter, Function Tables, Time Out and Minor Cycle that are used in the UNIVAC without the necessity of delving into its very complex construction. In this and the following chapters we shall make constant reference to these devices as they are used in UNIVAC. Their functions are essentially the same as in the binary computer of Chapter II and the reader should require little effort to make this extension.

The Supervisory Control Console, figure 1, is the principle point where the UNIVAC'S operation is monitored. Both operators and maintenance personnel are concerned with its function. Associated with the SCC is the Supervisory Control Printer, SCP, which is a standard Uniprinter dolly, and an oscilloscope. The SCP is used in the 50 m and 10 m orders while the oscilloscope is used primarily by maintenance personnel. The points of principle interest in SCC are the Supervisory Control Panel (SC Panel, Chart E in appendix), and the Supervisory Control Keyboard (SCK, Chart F in appendix).

This chapter will be a description of the Panel and SCK. Not all the features of each will be covered, just those of direct concern to the operator. As an aid in locating neons and switches on the Panel, Chart E is marked with a lettered and numbered grid. Also, the chart may be fully extended for ease of reference while reading this and the following chapters.

SECTION 2

Interrupted Operation

Under normal conditions, the UNIVAC is stopped by a programmed order. The instruction that does this is the 9 m. The manner in which the computer is stopped is shown in figure 2.

An activating signal produced by the 9 m instruction sets the Stop Flip-Flop. Now once the ending pulse sets the Time Out Flip-Flop, the Stop FF prevents it from being reset. As illustrated in Chapter II, the Time Out Signal blocks most of the gates in the computer, preventing any important action from taking place. Thus by preventing time out from being removed, the computer is effectively stopped.

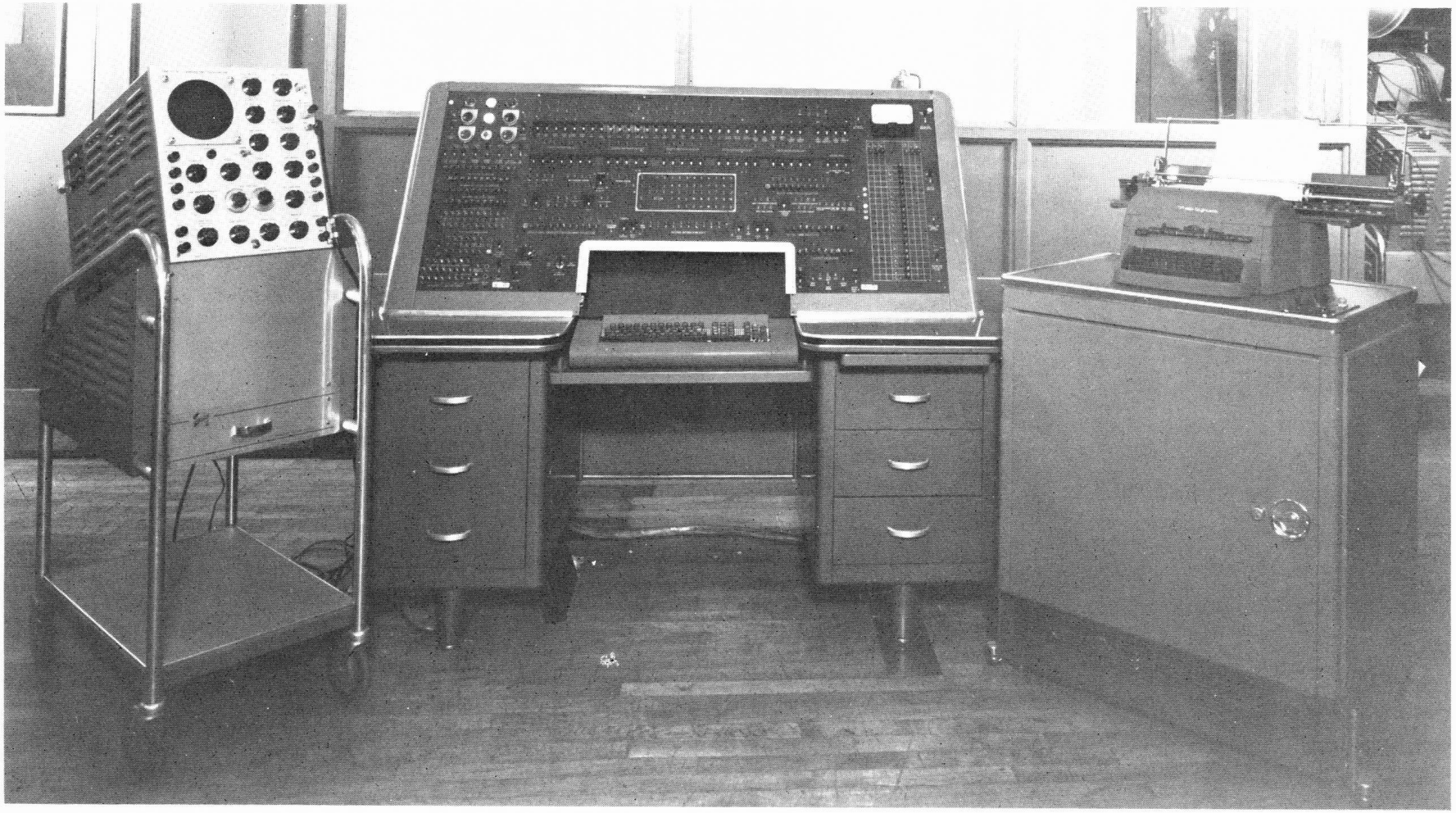


FIG. 1

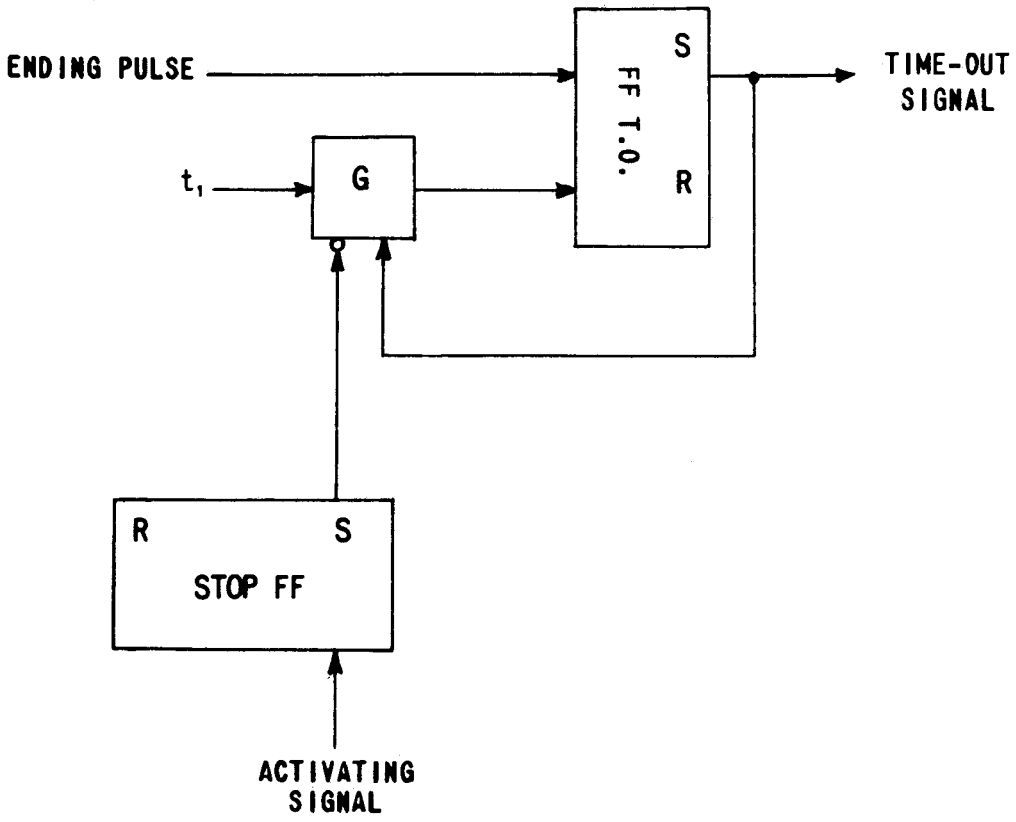


FIGURE 2

A neon on the SC Panel, F-5.2, will be lit whenever the Stop FF is set. The computer may be started again by resetting the Stop FF. This is done by depressing the Start Bar on SCK.

When the computer is stopped, the Stall Neon, F-4.3, will be lit. This neon is lit whenever the Cycle Counter has not been stepped at least once in three seconds. If the Stall Neon is lit but not the Stop Neon the UNIVAC is said to be stalled. This can occur in several ways, for instance, if a tape is rewound with interlock, 8n m instruction, and another tape order for this same Uniservo is to be executed, the computer is prevented by the interlock from executing that instruction and waits until the Interlock is removed manually. Only the Stall Neon will be lit in this case.

The Stop Switch, F-5.1, a non-locking switch, provides a manual method of setting the Stop and Time Out Flip-Flops. However the switch should not be operated unless the Stall Neon is lit.

Since the UNIVAC was designed as an automatic computer no positive method of manually stopping operation at an arbitrary time without causing an error was provided. A method of manually stopping the computer which has been found to be generally reliable is through use of the Interrupted Operation Switch, IOS, H.3-6.2.

The IOS is a five position locking switch. The operation of the computer with the switch in each of its five possible positions are:

1) IOS in NORMAL (as shown on chart) or CONTINUOUS position. UNIVAC operates directly under control of programmed instructions. This is the normal mode of operation.

2) IOS in ONE INSTRUCTION position. The computer is stopped after every stage of the Cycle Counter CY. Thus if the computer is executing the α stage for example and IOS is on ONE INSTRUCTION the α ending pulse that steps CY to β will also set the Stop FF. The computer then stops on β time out.

3) IOS in ONE STEP position. The UNIVAC is stopped after every stage of the Program Counter except when a V m, W m, Y m, Z m, or shift order is being executed.

4) IOS in ONE OPERATION position. The UNIVAC is stopped on every time out.

5) IOS in ONE ADDITION position. The UNIVAC is stopped on every time out and additional time outs are inserted during the repetitive stages of the multiplication, division, multi-word transfers, and shift orders.

The operator is concerned primarily with the CONTINUOUS, ONE INSTRUCTION, and ONE OPERATION positions of the Interrupted Operation Switch. The other two positions are provided for maintenance use.

As mentioned, the IOS may be used to stop UNIVAC from the SC Panel. By placing IOS in the ONE INSTRUCTION position* the computer generally stops with the last instruction properly executed.

* Any of the non-continuous positions may be used but ONE INSTRUCTION is generally the most convenient stopping point.

SECTION 3

Breakpoint Selectors

There is another way in which the UNIVAC may stop through a programmed operation. In the discussion of the Qn m and Tn m orders in Chapter III it was mentioned that if either of these orders are being executed by the computer and the Conditional Transfer Breakpoint Button n (n a number between 0 and 9) has been depressed, the computer will stop on time out of PC-2. The Conditional Transfer Buttons are shown at G.1-5 to 7.3. Ten buttons are provided for n equal to zero through nine. When one or more of these buttons have been depressed the computer will stop on executing the Q or T instruction whose second instruction digit corresponds in number to one of the depressed buttons. An eleventh button, the "all" button, will stop the computer on every conditional transfer order, irrespective of the second instruction digit.

When the computer stops the Conditional Transfer Neon, G-8.1, will be lit if the computer is going to transfer control, or it will be dark if the computer is not going to transfer. A switch, G.1-8.3, allows the result of the comparison to be altered. If the switch (non-locking) is lifted the computer is forced to transfer control on the next step, PC-2, but if the switch is depressed the computer will not transfer control, irrespective of the contents of rA and rL. The left-most breakpoint button, not labelled, is the release button. When it is depressed all "set" buttons are restored to their non-operative state. A set of neons, G-5.1 to 7.3, are provided for ease in determining at a glance what breakpoint buttons have been depressed. If breakpoint 6 has been set for example, the neon just above it will be lit.

In addition to the Q and T breakpoints the Comma Breakpoint (, m) order is occasionally used by programmers. When the Comma Breakpoint Switch, G.1-9.1, a locking switch, is depressed the computer will stop on every , m instruction.

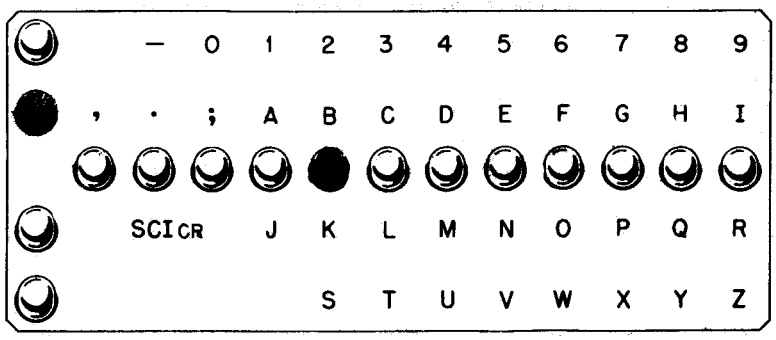
An option for stopping the computer on every print order, 50 m, is also provided. If the Output Breakpoint Switch, I.1-18.1, a three position locking switch, is in its normal position the 50 m order is interpreted as a type-out. If the switch is in the skip position every 50 m is ignored, while if the switch is in the breakpoint position the computer stops on PC-2 (TO) of each 50 m.

In all of the above cases, when the computer stops it may be started again by operating the Start Bar.

SECTION 4

Static Register

Just as in the binary computer, the Static Register in UNIVAC is the focal point of the computer's operation. It consists of a set of 27 flip-flops which hold the instruction currently being executed. Only the first, second, fourth, fifth, and sixth instruction digits are stored in SR. Since the third instruction digit is not used in defining the instruction it is not stored. Of those five instruction digits stored, only the check pulse and numeric parts of the second, fourth, fifth, and sixth are set-up in SR, while all seven binary bits of the first instruction digit are stored. Leads from the SR flip-flops are brough out to neons on the SC Panel so that the contents of SR are visible to the operator. These neons are in the center portion of the Panel, C.1-5 to 16.2. Each neon lit indicates a binary 1 while the neons that are dark mean binary 0. For ease in interpreting, the first instruction digit is decoded in the neon grid in the center of the SC Panel, D.3 to F.2-9.1 to 13.1. The first instruction digit in SR can be found at the intersection of the row and column having lit neons. For example, the digit decoded in figure 3 is the letter B.



NEON ON

NEON OFF

FIGURE 3

The second instruction digit is decoded in the horizontal row of neons, G-10 to 13. No decoding neons are provided for the three memory location digits (4th, 5th, and 6th instruction digits). The operator must learn to read the excess-three notation rapidly and accurately. Only a little practice will provide the necessary facility.

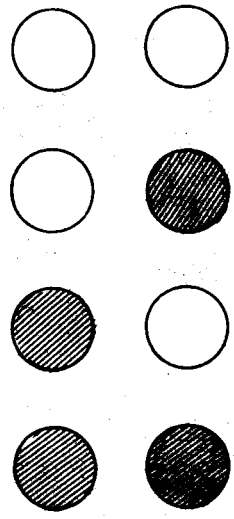
Just below the Static Register Neons is a set of Static Register Switches, D.1-5 to 16.2. Through these switches any desired instruction may be set up in SR. The switches may be locked into their appropriate position, though they are usually just lifted to set or depressed to reset the appropriate flip-flop, lighting its corresponding neon of course, and then returned to their normal position.

The three switches, D.1-17.1 to 18, are provided as aids in manually placing instructions in SR through the SR switches. These switches are selective in their action. The right-most switch when lifted places decimal zeros in SR, and when depressed inserts binary zeros (all flip-flops reset). The center switch affects the memory location flip-flops (instruction digits 4, 5, and 6). In the up position it clears them to decimal zeros, depressed it clears them to binary zeros. The left-most switch controls first and second instruction digit flip-flops. When lifted it jams decimal zeros into these flip-flops, and when depressed it jams them to binary zeros. These three switches are non-locking and thus return to their normal inoperative position after they have been moved.

SECTION 5

Control Circuit Neons and Switches

The fundamental mode of operation of the UNIVAC is the four stage cycle α, β, γ, δ. The Cycle Counter is a two-stage binary counter that registers the current stage of the cycle being executed. Two neons, F-7.3 to 8, are connected to CY and indicate its state:



α STAGE (BOTH NEONS OFF)

β STAGE (RIGHT NEON LIT)

γ STAGE (LEFT NEON LIT)

δ STAGE (BOTH NEONS LIT)

FIG. 4

CY may be cleared to the α state by operating the non-locking Clear CY Switch, F-7.1.

Some instructions must be executed in several steps, multiplication, addition, division, among many others. The Program Counter, PC, is a four stage binary counter that keeps track of the current step in the execution of an instruction. Four neons, E-8 to 9, indicate the state of PC. All neons dark signify PC-1 and all neons lit indicate PC-16. The Clear PC Switch, F-8.2, restores this counter to PC-1 when depressed.

At D-3.2 is a three position locking switch. In the "up" position it prevents the Control Counter from being changed. In the "down" position it prevents the Cycle Counter from being changed. The switch is disengaged when it is in the center, normal, position.

As an example of the use of the SC Panel switches and neons, a procedure for inserting an extra instruction into the computer follows:

Since instruction can be executed on γ or δ time only, it is necessary for CY to be in one of these states. Assuming that the computer has been stopped in some manner appropriate to the insertion of the instruction, we wish to execute an additional order without losing our place in the program. This procedure holds for all orders except 30 m, 40 m, 3n m, 4n m, 5n m, 7n m, Qn m, Tn m, or U m.

1) Place IOS in ONE INSTRUCTION position.
2) If CY does not read γ (10) or δ (11) operate Start Bar until it does.
3) Depress switch, D-3.2, to Retain Instruction position.
4) Set up desired instruction in SR by using SR Switches. It may be convenient to first clear SR to binary or decimal zeros.
5) Lift non-locking Clear FFTS Switch, D-2.2, and release.
6) Operate Start Bar. Computer will execute the instruction just entered in SR.
7) Un-retain instruction by resetting switch used in 3). Place IOS on CONTINUOUS. Operate Start Bar. Computer will execute programmed instructions.

SECTION 6

Output Selectors and Programmed Inputs

The nine buttons at H.1-16 to 18 are the Output Selector Buttons used in conjunction with the 50 m instruction. Under normal operation the button labeled M is depressed. Then a programmed print order, say 50 125, will print the word in memory cell 125 on the Supervisory Control Printer. Should one of the following buttons be depressed a 50 m order will operate as noted:

Button Depressed	Action
A	(rA) ⟶ SCP
X	(rX) ⟶ SCP
L	(rL) ⟶ SCP
F	(rF) ⟶ SCP
C	(Control Counter) ⟶ SCP
CR	(Control Register) ⟶ SCP
SYI	(Synchronizor Input Register) ⟶ SCP

A different action takes place when the Empty Button is depressed. This will be covered in the following chapter.

The Block Sub-Divider Buttons, E.1-14.2 to 17, and their neons just above, are used to control the mode of writing on magnetic tape. For certain auxiliaries, the High-Speed Printer and the Tape-To-Card Converter, the input tape must have a space between each ten word item. When one of the numbered Block Sub-Division Buttons is depressed, a write order for that same numbered Uniservo is interrupted for a short instant after each ten words have been written. Thus a block of sixty words is broken into six sets of ten words each. A space of approximately one inch is automatically inserted between each set of ten words for High-Speed Printer use (Buttons 1 to 7). A space of approximately one-tenth inch is automatically inserted between each set for Tape-To-Card Converter use (Buttons 8, 9, -). When one or more buttons have been set their corresponding neons are lit. The left-most unmarked button resets all set buttons.

When a 10 m instruction is executed by the computer it lights the Input Ready Neon, I.1-16.2, and then waits until the operator types in one word from the Supervisory Control Keyboard, SCK. Any of the letters of the alphabet, numbers or punctuation symbols are permissible characters, the particular ones required are specified by the program being run. When the twelfth digit has been typed (digits are numbered from left to right) the 12th Digit Neon, I-17, will light. The operator then strikes the Word Release Key which completes the transfer of the typed word into the computer and allows the UNIVAC to continue under programmed control.

As each character is typed into the computer it is printed on SCP. Operating the Word Release Key causes a period to be printed--the printer switch must be set to the Computer Digit position. This allows a visual check on the word typed. Should a wrong digit be typed and this fact recognized before the Word Release Key is struck, the whole typing may be restarted afresh by striking the

Erase Key. Operation of the Erase Key causes the printer to line feed and return the carriage to the left margin stop.

The Input Error Neon immediately below the 12th Digit Neon, will be lit under any of the following conditions:

1) Two or more keys struck simultaneously.
2) The Word Release Key struck before the 12th digit has been typed.
3) Any key except the Word Release Key operated after the 12th digit has been typed.

In any of the above events the typing may be redone after striking the Erase Key.

SECTION 7

Miscellaneous Neons and Switches

In addition to the switches and neons discussed in Section 5, there are other control circuit indicators on SCP. These are generally of much less significance in normal operation than those already discussed.

High Speed Bus Speaker: Since UNIVAC is electronic in its construction it is essentially silent in operation. An audible signal is provided by means of the High Speed Bus Speaker. There are three possible modes of operation controlled by a switch at I-4. When this switch is in the

1) Normal position, the speaker is non-operative.
2) Raised position, an audible signal is produced whenever information passes over the High Speed Bus.
3) Lowered position, the speaker emits an audible signal only when the computer is stalled (CY not stepped in at least three seconds).

Just to the right of this switch is the Speaker Volume Control Knob. Usually the speaker is operated in mode 3, but when de-bugging routines, useful information can be imparted to the trained programmer when the speaker is in mode 2, in particular, closed loops (un-ending iterative cycles) emit a characteristic sound.

Overflow Neon: This neon at G-9.3, is lit whenever an overflow (on A m, X m, S m, or D m instructions) occurs. This is of no importance in itself, unless the second instruction digit is a minus sign (A- m, X- m, S-m, or D- m). If overflow occurs on these instructions the computer stops on the next time out minor cycle. Further computation will proceed at the direction of the programmer. Often the programmer may wish to ignore the overflow and continue with the program. In this case the operator must execute certain steps; but first it will be helpful to recall the action of the computer when overflow occurs. The effect of an overflow is always on the next α and β stages. The normal occurrence in α time is to transfer the CC reading to SR. However, if an overflow has occurred on the previous γ or δ, this transfer is blocked

and decimal zeros from the Cycling Unit are sent to SR instead. This means that the following β time selects the instruction pair in cell 000 to be sent to CR. Also, the addition of 1 to CC is blocked. After executing the instructions in 000 the following stages are executed normally. The instructions being executed are the ones following the order pair in which the overflow occurred--unless a transfer of control was one of the instructions in 000. Now, assuming the computer has stopped on an overflow and the programmer wishes to ignore this:

1) Put IOS on ONE INSTRUCTION. Operate Start Bar.
2) If the overflow occurred on δ time the Cycle Counter will now read α. But if overflow occurred on γ time CY will read δ. In this case operate Start Bar once to execute right hand instruction and step CY to α
3) Operate Start Bar twice. CY will now read γ and the Overflow Neon will be out.
4) Clear CY to α by using the Clear CY Switch, and place IOS on CONTINUOUS. Operate the Start Bar and computer will continue with the program.

Time Out Neon: This neon, located at F-6, is lit whenever the Time Out Flip-Flop is set.

Time Selection Neon: This neon, located at F-6.1, is lit whenever the Time Selection Flip-Flop is set.

SECTION 8

Checking Circuits

In the descriptive computer of Chapter II we postulated a basic pulse repitition rate of 1,000,000 pulses/second. This may have seemed quite large but in UNIVAC this rate is 2¼ times larger. This means that in an eight hour work day UNIVAC generates 64,800,000,000 pulses. Your brief introduction to programming in Chapter I has demonstrated how essential each pulse is for correct calculation. Suppose in the second example of that chapter a single pulse is lost in transferring a word into rL on line 003. This could make the Q m instruction operate incorrectly, transferring control before the final sum was obtained. Just one pulse in error among the 2¼ million generated every second can make the answers to any calculation wrong not merely by one part in 2¼ million, but completely incorrect. Even though the computer would have operated correctly 2,249,999 times and failed only once the answers would be meaningless. The principles of digital computation require absolute accuracy if the results are to be reliable.

There are three basic checking methods used in UNIVAC: Odd-Even, Duplicated Circuits, and Logical Checks. In every case* these methods serve only to detect the occurrence of an error and prevent any further computation which would propagate the error. Correction of the error is then a manual task.

* An exception is the Automatic Re-Read device.

Errors may be classified as either central computer errors or input-output errors. Their effect and methods of correction are sufficiently different to warrant this classification.

Central computer errors comprise all errors produced by the 1000 words of main memory; the special registers rA, rX, rL, rF, rV, rY, rI, CC, CR, and SR; the arithmetic circuits; and the control circuits. Generally these errors prevent the resetting of time out which stalls the computer in the next time out period, but some of the control circuit checks prevent further execution of the instruction beyond the stage in which the error occurred.

Input-output errors, as the name implies, covers those directly produced by reading or writing on tape or through the Supervisory Control Printer. The occurrence of an error in these cases prevents the computer from doing another order of the same type or of using the same Uniservo. For example, if an error was made on this order, 12 000 (fill rI with a block from the tape on Uniservo 2), another 1n 000, 2n 000, 3n m, 4n m, 30 m, or 40 m instruction would stall the computer. A 52 m, 72 m, 62 m, or 82 m would also cause a stall, but not a 53 m.

SECTION 9

Duplicated Circuits

The reader has probably noticed during the discussions of the Supervisory Control neons and switches that many seem to be duplicated. Most of the control and arithmetic circuitry in UNIVAC is duplicated.

The High Speed Bus, HSB, links the memory and special registers. A duplicate, $\overline{\text{HSB}}$, is provided and the information on each bus is constantly being compared for identity. Should they fail to register identical contents an "error" flip-flop is set which inhibits the resetting of time out. The High Speed Bus Comparator Neon, A.2-6.2, is lit whenever this error flip-flop is set. Immediately below this neon is a switch which may be set to disable the HSB Comparator Checker.

The following registers and circuits are also duplicated. The duplicate parts being referred to as the "barred" circuits. The barred and unbarred registers are continually compared for identity in every pulse position and the barred and unbarred circuits compared for identical outputs when they are being used. An appropriate error flip-flop is set whenever they register different results. The barred circuits and registers are connected to $\overline{\text{HSB}}$. The Error Delete Switches are below the neons:

duplicated parts	location of comparator error neon
rF	A.2-8.1
rL	A.2-8.3
rA	A.2-9.1
rX	A.2-9.2
Adder output	A.2-8
Adder input, minuend	A.3-6.3
Adder input, subtrahend	A.3-7.1

Comparator (conditional transfer)	A.2-10
Time Out FF	A.2-13
Cycling Unit	A.2-13.1

Two error neons are shown. The upper neon is lit when the duplicate circuits disagree, the lower is lit when a further logical check is not met.

The following circuits are duplicated as well, but are not provided with an error neon or delete switch. However neons indicating the state of each circuit are on the SC Panel, the barred circuit neons directly below the unbarred:

Program Counter
Cycle Counter
Stop FF
Time Selection Circuits
Multiply-Divide Circuits

There is further duplicated equipment in the UNIVAC which is not accessible from SC Panel.

SECTION 10

Odd-Even Checks

The second checking means employed in UNIVAC is the odd-even check. There are five odd-even checkers in the UNIVAC, which count the number of binary ones of every character passing through the checker. Whenever an even count is registered for any character an error flip-flop is set. This then is the reason for inclusion of a check pulse position in every character--so that the sum of the binary ones for valid digits is an odd number. The location of the checkers are:

High Speed Bus Odd-Even Checker. This is a duplicated checker. Every character passing over the duplicated HSB is checked. This includes all transfer operations to and from the memory and between special registers. Should an even count be registered for any character the error flip-flop is set, lighting the neons at A.2 and A.3-6.

Adder Minuend Odd-Even Checker. The error neon is at A.2-6.4. This checker passes on the odd-even count of every digit entering the unbarred minuend input of the duplicated adder.

Adder Subtrahend Odd-Even Checker. Neon at A.2-7.1. The function of this checker is similar to that of the minuend checker, but operates on the other input to the adder.

Input Synchronizer Odd-Even Checker. This is a duplicated checker, with the error neons at A.2-15.3 and A.3-15.3. This device checks the odd count of every character read from magnetic tape into rI, and from SCK into SYI-1 (see Chapter IV).

Output Synchronizer Odd-Even Checker. The error neon of this checker are located at A.2-16. The checker counts the oddness of every character being written on tape or SCP. It is located beyond the last vacuum tube stage in the Uniservo circuits providing the maximum assurance that the character is correctly written at that time.

In addition to the O-E checks on transfers, the regular operation of the computer is interrupted every five seconds when CY reads α for the Periodic Memory Check, PMC. During PMC the contents of the memory are read into the HSB O-E Checker. Should an even count be registered for any of the 12,000 characters the HSB O-E Checker will set the error flip-flops and stall the computer during the next time-out period. The channels (ten word groups) that contain incorrect digits may be determined from the SC Panel (this is covered in Chapter VII). PMC prevents a faulty channel from going undectected for long periods of time and possibly dropping enough pulses to pass the odd-even check. The time necessary to perform PMC is included in the average instruction times. A switch at D-3 may be set to inhibit performing PMC.

SECTION 11

Logical Checks

In addition to the duplicated circuits and odd-even checkers there are a large number of internal logical checks designed to further insure error free computation. Logical checks are employed wherever it is not feasible to duplicate equipment or where no data transfer is involved to make use of odd-even checks.

Tank Selector Checker. This checker assures us that the proper memory channel is selected, and thus is a check on the fourth and fifth instruction digit set-up of SR; the error neons are at A.2 and A.3-14.2. A further word of explanation is necessary for this checker. There are two general types of checking circuits:

1) In a negative checker the error flip-flop is set when an error is detected.

2) In the positive type checker, the error flip-flop is set first and only correct operation will reset the flip-flop in time to prevent stalling the computer.

This is a positive type checker. If the upper neon is lit and the computer is stalled it means the fourth instruction digit was set up incorrectly, if the lower neon is lit and the computer stalled it indicates that the fifth instruction digit is incorrect. It is quite possible to have a Tank Selector Error through a faulty program. An instruction B al2 for example, will show a fourth instruction digit error since there is no memory channel al. The sixth instruction digit set up is checked by the Time Selection Circuits which compare the check pulse in SR against a computed check pulse.

Function Table Intermediate Checker. The error neon is located at A.3-14. This neon is lit whenever an improper combination of lines is fed into the decoding function table. It is thus a check that the first instruction digit

was set up correctly in the Static Register. This checker also acts as a shift selector check, making sure that one and only one line is picked up by the second instruction digit decoding function table when a shift order is being executed.

Function Table Output Checker. This is a duplicated positive type checker, the error neons are at A.2 and A.3-13.3. In addition to the function table signals required by an instruction, dummy lines are attached to the table so that each order will select an even number of signals.

Tape Check. As pointed out in Chapter IV, along with the seven information bits recorded on tape for each character, an eighth bit called the Sprocket Channel Pulse is also recorded. When information is being read from tape the Sprocket Channel Pulse indicates the presence of a character and actually initiates the process of synchronizing the incoming information with the timing of the computer. If a Sprocket Channel Pulse is not read from tape along with information pulses the Tape Check Error flip-flop is set. The error neon is at A.2-16.3.

Input-Output Interlock Checker. The error neon is at A.2-16.2. The input-output interlock circuits, you will recall, are set at the beginning of any input-output operation and are reset when that operation has been completed successfully. When the interlocks are set further orders of the same type or using the same equipment are prevented from being executed. It is essential to correct operation of the computer that the interlock circuits function properly. This checker, when set, indicates one of the following failures:

1) The read interlock failed to set at the beginning of the last read order.

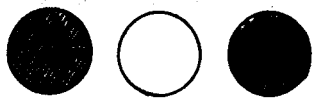
2) The write interlock failed to set at the beginning of the last write order.

3) The Uniservo in question was set to execute a backward read when a forward read was ordered.

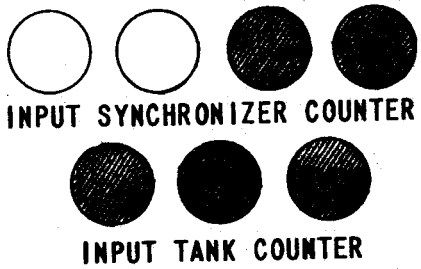
Input Synchronizer > 720 Checker. The error neon is at A.2-15.1. As mentioned in Chapter IV, digits are recorded serially along the tape, and are thus picked up one at a time when the tape is read. The computer counts the number of digits read and after the 720th digit (last digit of the 60th word) has been read and the space between blocks is encountered the read is terminated. Through a failure in the input-output control or photocell circuits a short (less than 720 digits) or a long (greater than 720 digits) block may be encountered. Either of these two cases sets the > 720 error flip-flop.

As far as the computer is concerned, a short block is defined as a read of 59 complete words and at least one, but less than twelve more digits, followed by the space between blocks. The Uniservo will then stop reading tape and set the > 720 error. A long block occurs when the computer reads a full 60 words and at least one more digit before encountering a space between blocks. The tape stops in the next space between blocks or photocell area, whichever is first, and sets the > 720 error. In either case setting > 720 error prevents the next read order or Supervisory Control input from being executed or any other order affecting the Uniservo causing the error.

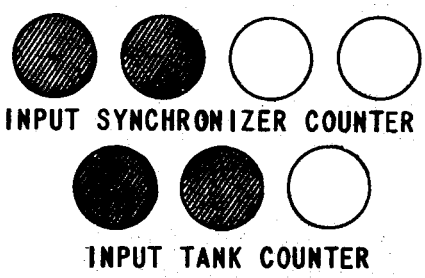
The Input Synchronizer and Input Tank Counters, F to F.1-14 to 14.3, are used in deciding which 720 error occurred, long or short block. The Input Tank Counter counts in the excess one system. For example, if a shaded circle indicates a lit neon, the ITC reading of the configuration shown is 4:



The Input Synchronizer Counter counts in normal excess three. Together both counters indicate the number of complete words assembled plus one, the count starting from 00. Thus after a correct read has been executed the ITC should read 7 (an XS-1 6) and the ISC should read an XS-3 zero:



Bearing in mind the definition of a short block, if the computer should stall with a >720 error registered and if ITC and ISC read 59:



a short block is the probable error. Any other counter reading probably implies a long block. This rule in itself is not always a sure method of differentiating the two types of error, but is sufficient for most cases.

CHAPTER VI

NORMAL OPERATING PROCEDURES

SECTION 1

Introduction

This Chapter will describe efficient UNIVAC operating techniques. The two goals of efficient operation are speed and accuracy. The beginning operator should strive for careful and accurate operation. Hours of valuable computer time may be lost through the errors of an inexperienced "speedy" operator.

A step by step procedure will be listed for all the most common Supervisory Control operations. What to do in each normal operating procedure is the subject of this Chapter.

Only brief mention is made about why specific switches are used in a definite sequence.

Before the operator performs any manual operation on the SC Panel he should first check to see that:

1. The IOS, (the five position switch on the bottom left section of the SC Panel) is NOT on Normal

2. The STOP neons are lit: (Depress stop switch to light stop neons)

The operator must memorize the excess 3 binary code.

The binary code patterns for the numbers 0.....9., as they will appear in the Static Register Neons, are:

First Instr. Digit	2nd, 4th, 5th, 6th Instr. Digits	Decimal Number
1000011	10011	0
0000100	00100	1
1000101	10101	2
1000110	10110	3
0000111	00111	4
0001000	01000	5
1001001	11001	6
1001010	11010	7
0001011	01011	8
1001100	11100	9

The binary code patterns for letters do not have to be memorized, because they are used less frequently and when needed can be encoded from the first instruction digit decoding matrix in the center of the Supervisory Control Panel.

SECTION 2

Interrupted Operation Switch (IOS)

The IOS is a five position switch located on the Control panel just above and to the left of the SCK.

The position of the IOS is the first thing an operator should check before doing any manual intervention in the UNIVAC's normal, programmed, automatic, operation. Its function in each position is reviewed below:

The five positions are:

1. NORMAL (perpendicular) This position allows the computer to run continuously under control of the instructions stored in its memory. Once the computer is started with the IOS on NORMAL, it will stop only upon decoding a stop or breakpoint instruction in its memory. (The UNIVAC may STALL as a result of a programming error, an operational error, or a computer error detected by the built-in checking circuits, but it is stopped only when the Stop FF's are set - neons lit next to Stop Switch, left side of control panel.)

2. ONE INSTRUCTION - (down) This position causes the computer to stop at the beginning of each stage of the four stage cycle. It will stop UNIVAC on alpha time out, beta time out, gamma time out and delta time out periods.

3. ONE OPERATION - (up) - on this setting the computer will stop on every Normal time out period. This is the most useful position of the IOS for doing manual operations.

   On the following instructions, computer operation is exactly the same on both the ONE INSTRUCTION and the ONE OPERATION position of the IOS:

B m	H m	U m	Zm	00 m
C m	J m	V n	;n m	90 m
E m	K m	W m	.n m	
F m	L m	X m	-n m	(Alpha, Beta)
G m	R m	Y m	on m	

   The remaining UNIVAC instructions (except multiplication and division) are all multi-stage instructions that make use of the Program Counter to identify each stage. There is a normal time out period introduced between each PC stage of these multi-stage instructions and when the IOS is on ONE OPERATION, the computer will stop not only on alpha, beta, gamma, and delta time outs, but also on these time out periods that come between the successive stages of the multi-stage instructions.

4. ONE STEP - (left) This position of the IOS is seldom used in the normal operation of the computer. It acts the same as the ONE OPERATION position on all instructions except the multiplication, division, and shift instructions. The multi-stage instructions do not

have normal time out periods between all of their successive Program Counter stages. The ONE STEP setting will cause the computer to stop whenever the Program Counter is stopped, forcing time out periods between successive stages where normally there are none.

5. ONE ADDITION (right), the last position of the IOS will control the UNIVAC exactly as the ONE STEP position does, except that it will also cause the computer to stop after each addition during the repeated stages of multiplication and division. Both the ONE STEP and the ONE ADDITION position of the IOS are used mainly to aid in engineering trouble shooting. They are of little value to the operator under normal running conditions.

All of the UNIVAC instructions except the U, Q, and T can be set manually into the Static Register while the computer is stopped and then executed by the operator's hitting the Start Bar.

The U, Q, and T instructions must be entered into the Control Register before they can be properly executed (See SCI_{cr}).

If the 30 m, 40 m, 3n m, and 4n m instructions, the only ones that involve transfer of the contents of register I to sixty consecutive memory cells, are to be executed by manual set up in the static register, the IOS must be on ONE OPERATION and the Static Register must be manually set six times to complete the instructions properly. The same multiple manual set-up requirements apply to the 7n m and 5n m, the write on magnetic tape instructions. For example, to perform manually a 72 940 instruction, the six required set-ups of the Static Register are:

72	940
72	950
72	960
72	970
72	980
72	990

The operator must remember that the instructions in the Control Register are altered by the above manual operation.

A better method of performing manually the input and output instructions described above is to enter them directly into the Control Register (see SCI_{cr}). When this is done the proper multiple set up of the Static Register will be taken care of automatically.

The U, Q and T instructions can be executed manually only by first entering them into the Control Register (See SCI_{cr}).

SECTION 3

Turning on UNIVAC

Most of the switches, lights and buttons referred to below are on the upper left hand section of the Supervisory Control Panel.

1. Set Servo Power Switch, B.2-12.2, to OFF (up) position.
2. Power ON - The orange control power lamps B.1-3 will be lit when the UNIVAC is connected to an outside power source.

A series of switches and circuit breakers not on the Supervisory Control Panel must be closed to connect power to the computer. Check with the engineer in charge to learn the location and restrictions on the operation of these switches.

3. Stand-by Power Switch, B.3-1.1, set to the ON or "Restore" position.
4. Push Heaters ON button.
5. Wait until YELLOW Ready Lamp is lit (approx. 6 min.)
6. Turn Key in the DC Lock, C-3, clockwise to the ON position.
7. Push DC ON button.
8. IMMEDIATELY Operate the following switches to start the Cycling Unit:

    1. Depress and release the CU-TSC Clear Switch, D-1.1.
    2. Depress the CU Start Interlock Switch, D-1.3, and hold down.
    3. Depress the CU Start Switch, D-2 and hold down.
    4. Release the CU Start Interlock Switch.
    5. Release the CU Start Switch.

If the above steps have been executed correctly, a single pulse will now be circulating in the Cycling Unit generating the timing signals necessary to control all internal operations of the computer. If the UNIVAC has been turned off for a long period of time it will take about 30 minutes to bring the mercury up to operating temperature; since the timing signals are also used to control the temperature of the mercury memory tanks.

The following operations cannot be performed successfully until all mercury storage is at operating temperature.

INITIAL CLEAR OPERATION

1. Lift the Initial Clear 1 Switch and hold up, D-1.3.
2. Lift the Initial Clear 2 Switch and hold up, D-2.
3. Release the Initial Clear 1 Switch.
4. Release the Initial Clear 2 Switch.
5. Depress General Clear Switch, D-2.2, then lift same switch to Clear FF TS, and release.

The above operations have erased all of the one word registers, except registers A and $\overline{A}$. Also, register V, the 2-word register, is erased. Registers A and $\overline{A}$ will both contain the word: (000000 000005). The General Clear Switch sets a host of binary memory devices (Flip-Flops and Binary Counters) to an initial state.

MEMORY CLEAR

Next fill the mercury memory with decimal zeros.

1. Depress the Memory Clear Switch, D-3. It will lock down.
2. Set the IOS on ONE OPERATION.
3. Depress and release the Clear C Switch, G-14.1. Note that the PMC Neon, A.2-5.3, will be lit at this time.
4. Hit the Start Bar several times and observe the fifth instruction digit neons in the Static Register change. (Should increase by one with each step.)
5. Set IOS on ONE INSTRUCTION.
6. Hit Start Bar.
7. Lift the Memory Clear Switch to the Inhibit PMC position.
8. Depress the I.F. BIAS CONTROL Switch B.3-4.1, which will light the neon directly above it.

Note at this time, that the PMC Neon will be out, the Static Register will contain all binary zeros, the Cycle Counter will be alpha, and the Time Out and Stop Neons will be lit.

MEMORY CHECK

To see that the memory clear operation was performed successfully, the operator should next let the UNIVAC run on the skip instructions in the memory:

1. Set the IOS on ONE OPERATION.
2. Depress and release the Clear C switch.
3. Set IOS on Normal.
4. Operate Start Bar.

If the memory contains all decimal zeros, the computer will <u>stall</u> -- light the Stall Neon, F-4.3 - after approximately $4\frac{1}{2}$ seconds. The Static Register will contain 01 000, the Cycle Counter will be on beta stage, the Time Out Neons will be lit and the FT Intermediate Checker Neon, B.2-14 will be lit.

(If the memory check operation does not terminate as above, this probably means that the mercury has not as yet reached proper operating temperature. Wait a few minutes, then repeat the <u>Memory Clear</u> operation.)

5. Depress the Stop Switch, F-5.1
6. Set IOS on ONE OPERATION
7. Depress Clear C Switch

Now the UNIVAC is ready to start a problem.

SECTION 4

Starting a Problem

1. Mount tapes as required.

   Check to see that there are snap rings on data tapes and instruction tapes. Also make certain there are NO snap rings in the tapes that will receive output.

   Check the Uniservo plugboard in the corner of the Central Computer nearest the Uniservos to see that the Uniservos are numbered in accordance with the plugboard. (Sec. 6 Chapter IV).

2. Turn on SERVO POWER switch, B.2-12.2.

3. Set up the Supervisory Control Printer as required.

   If no specifications are given, then set the left and right margins for a 39 digit line, and set the two position selector switch on the right of the typewriter keyboard to the COMPUTER DIGIT position.

4. Designate the Uniservos that are to receive output for the HIGH SPEED PRINTER or TAPE-TO-CARD CONVERTER by depressing the appropriate BLOCK SUB-DIVIDER BUTTONS. Uniservos numbered 1-7 are modified to record output for the High Speed Printer, while the Uniservos numbered 8, 9 and 10 are modified to record output for the Tape-To-Card Converter. The minus button, G.1-17.3, selects the 10th Uniservo for Tape-To-Card Output.

5. Set the required BREAKPOINT switched and buttons. (See Chapter V)

INITIAL READ OPERATION

6. Set the IOS on ONE INSTRUCTION.

7. Select the instruction tape by depressing the proper Initial Tape Selector Button, G.1-15.1 to 17.1.

8. Depress Output Selector Button M, H.1-16.

9. Depress the IS and OS Error Clear Switch, B.2-18.

10. Depress CLEAR I and O Switch, F-15.1.
    This erases the 60-word input and output registers.

11. Depress and release the Initial Read Switch G.1-13.1.

    Note the IR TIMES Neon, G-13.3, will come on and remain on for about 15 seconds. During this time one block of information immediately in front of the reading head of the servo selected for

initial read will be transferred into the sixty memory cells, and the Control Counter will be cleared to decimal zeros.

Another method of performing the INITIAL READ operation without using the built in INITIAL READ circuits is:

1. With the IOS on ONE INSTRUCTION, type into the Control Register (See SCI_{CR}) the word 1N0000 300000, where N is the number of the initial servo.

2. Depress and Release Clear I and O switch.

3. Execute the 1N and 30 instructions by hitting the Start Bar twice.

4. Set IOS on ONE OPERATION.

5. Depress and release Clear C switch.

Clear C Switch

Operating the Clear C Switch when the IOS is on NORMAL will put garbled data into the Control Counter, and light error neons on the top panel of the Supervisory Control. When the IOS is on ONE INSTRUCTION and the PMC Neon is lit, operating the Clear C switch will also result in errors. It is good operating practice always to set the IOS on ONE OPERATION before hitting the Clear C Switch.

SECTION 5

Supervisory Control Output Operations

There are two kinds of Supervisory Control Output operations:

1. Printing a single word from the memory or from any of the one word registers.

2. Printing a sequence of words from consecutive memory locations.

The 50 m instruction calls for printing out of memory cell m, but the address m will be ignored if any one of the Output Selector Buttons other than M is pushed.

To execute the 50 m instruction manually:

1. Set IOS on ONE OPERATION.
2. Step Cycle Counter to γ or δ
3. Check to see that M Output Selector Button is pushed.
4. Set up a 50 m instruction in the Static Register.
5. Operate Start Bar Twice and the word will print out.

If it is desired to print more than one word, after printing the 1st word on ONE OPERATION the Static Register will still contain the 50 m instruction just executed. Simply change m to the new memory address and:

1. Depress Clear PC switch (F-8.2).
2. Operate Start Bar twice and the new word will print out.

To print out the contents of any of the one-word registers:

1. Depress the appropriate Output Selector Button. Button A for Register A, etc.

2. With the Cycle Counter on γ or δ , perform the 50 m instruction as previously described.

It is always good practice to depress the M Output Selector Button after using the buttons to print out of the one-word registers.

EMPTY

The EMPTY Button (H.1-18) is used to print the contents of consecutive memory cells. The address of the words to be printed are specified by the Control Counter.

1. Set IOS on ONE OPERATION.
2. Depress Clear C switch if printing is to start with memory location 000. If printing is to start at any address other than 000, first set the Control Counter to the desired address. (See SCI_{cr}).
3. Depress the Empty Button, H.1-18.
4. Set IOS on NORMAL.
5. Operate Start Bar.
   When the required number of words have been printed:
6. Set the IOS on ONE INSTRUCTION.
7. Depress the M Output Selector Button.

SECTION 6

Supervisory Control Input Operations

There are three kinds of Supervisory Control Input Operations:

1. To enter a single word into a selected memory location. (10 m)

2. To enter a series of words into consecutive memory locations beginning at a selected location. (Fill)

3. To enter a single instruction word directly into the Control Register. (SCI_{cr})

The first kind of Input operation may be either programmed or manual. To execute a UNIVAC programmed type-in instruction note that the Input Ready Neon, I.1-16.2 is lit, the computer is stalled, and the Static Register displays a 10 m instruction. If there is no manual change to be made in the memory address m, simply type on the Supervisory Control Keyboard the 12 characters that belong in cell m. Note the 12th Digit Neon, I-17, will light indicating a complete word has been typed. Follow this by hitting the Word Release Key.

This will send the word just typed from an input storage register (SYI-1 or SYI-2) into the specified memory address and then release the computer to continue on with its programmed instructions. This operation is performed with the IOS on NORMAL.

If an incorrect key is struck in the type-in operation, a correction is made by hitting the Erase Key and retyping the complete word. Corrections must be made before hitting the Word Release Key, otherwise the incorrectly typed word will actually go into the memory.

The Erase Key causes the SC typewriter carriage to line space and return to the left margin.

The Word Release Key prints a period immediately after the 12 typed characters to identify them as a word entered into the computer from SC.

If the Input Error Neon (I.2-17) is lit during a type-in, this indicates that two keys were struck simultaneously or too quickly in succession, or that an incorrect number of characters was typed before hitting the Word Release Key.

Correction for this type of error is made as before, by operating the Erase Key and retyping the complete word correctly.

To enter a single word not called for by a program into the computer memory proceed as follows:

1. Set IOS on ONE OPERATION

   If Cycle Counter is not on γ or δ, operate Start Bar to make it so.

2. Depress Clear PC switch (F-8.2).

3. Set up a 50 m instruction in the Static Register.

4. Hit Start Bar twice and the contents of the memory cell to be changed will be printed out. This is done so as to positively identify the word in the memory that will be erased and replaced by the new word about to be typed. This is a check that the operator has set up the correct address, m, in the Static Register.

5. Change the 1st instruction digit in the Static Register from 5 to 1.

6. Depress Clear PC switch.

7. Set IOS on ONE INSTRUCTION.

8. Operate Start Bar and Input Ready Neon will light.

9. Type 12 characters and hit Word Release Key.

SCI_{cr} - Supervisory Control Input to Control Register

The SCI_{cr} operation allows the operator to enter a pair of instructions directly into the Control Register.

To perform this operation:

1. Set IOS on ONE INSTRUCTION
2. Lift Input Switch (I.1-16) up to the SCI_{cr} position.
3. Depress and release SCI_{cr} Switch, G-14.2. Now the Cycle Counter will be set to β and the Static Register will hold the SCI_{cr} instruction in the 1st instruction digit.
4. Hit start bar. Note that the Input Ready Neon is lit calling for the operator to type 12 digits on the keyboard.
5. Type a word containing two instructions and hit the **Word Release** Key. (See 10 m - Programmed Input)

   Note that the left hand instruction just typed will be recorded in the Static Register, the Cycle Counter will be on γ , and the Time Out FF's are set.
6. Drop the SCI_{cr} switch down to Normal.

The most frequent use of the SCI_{cr} operation is to change the contents of the Control Counter. For example, the operator is instructed to start a routine in memory location 325, the instructions being already stored in the memory. To do this, he operates the SCI_{cr} switches as above. (steps 1-5, typing in 000000000325). Now by hitting the Start Bar twice he executes first the skip instruction, then the U 325 instruction which changes the Control Counter to 325.

FILL

The purpose of the Fill Switch I.1-16 is to facilitate the entry of a series of words typed from the Supervisory Control Keyboard into consecutive memory locations.

1. Set the Control Counter to the address of 1st word in series by the Clear C Switch if the 1st address is 000, or by SCI_{cr} operation, is the 1st address is other than 000.
2. Depress the Fill Switch I.1-16. It will lock down.
3. Set IOS on NORMAL.
4. Operate Start Bar - Input Ready neon will light. Note at this time the Cycle Counter will be on β , the Program Counter will be on stage 2, and the Static Register will

contain the memory location into which a word will by typed.

5. Type a word on the Supervisory Control Keyboard and hit Word Release Key. This word will be entered into the cell specified by the Static Register, and the Input Ready Neon will immediately light again calling for the next word to be typed. Note that the address in the Static Register will be one greater than it was while the previous word was being typed. Continue to type each word until one less than the desired number have been entered.

Then:

6. Set the IOS on ONE INSTRUCTION

7. Type the **last** word and hit the Word Release Key.

8. Lift the Fill switch to its normal position.

CHAPTER VII
ERROR DIAGNOSTIC
PROCEDURES

SECTION 1

Introduction

In Chapter V the error detection circuits were described and the error neons on the SC Panel located and identified, while Chapter VI listed the SC procedures required in the normal operation of the UNIVAC. The purpose of this final chapter is to present those SC operations which may be used to diagnose and correct the effect of momentary component failures which might be encountered.

Because of the extremely thorough use of error detection circuits and the infinite variety of program coding, not all of the possible conditions which may be obtained can be treated. Indeed, the general procedure after occurance (and repair, if necessary) of a component failure would be to institute the general rerun procedure which every properly prepared program should include. However, diagnosis of what instruction was executed improperly is of material aid to the maintenance personnel and can best be provided by the operator. Again, many error conditions may be caused by only intermittent part failures of low frequency, capable of quick correction through use of certain SC procedures.

SECTION 2

A General Procedure Used In Central Computer Errors

Central computer errors are defined as those which stall the computer in the next time out period following commission of the error or which prevent further execution of the order beyond the step in error. When such an error occurs the operator may not see the instruction which was executed incorrectly set up in the SR Neons.

As noted in Chapter III some multi-Program Counter stage instructions have intermediate time out periods. For example, an HSB O-E error occurring in PC-1 of the multiply omder would stall the computer in the time out period of PC-2. If the instruction pair were M 433 X 000 the operator would see:

HSB O-E Error Neon	--- On
Time Out Neon	--- On
Stall Neon	--- On
Program Counter	--- 0001 (PC-2)
Cycle Counter	--- 10 (γ)
Static Register	--- M 433

However, if an Adder Subtrahend Error occurred in PC-15, the last PC stage of multiplication, the operator would see:

Adder Subt. Error Neon --- On
Time Out Neon --- On
Stall Neon --- On
Program Counter --- 0000 (PC-1)
Cycle Counter --- 11 (δ)
Static Register --- X 000

On all single Program Counter stage instructions or those with no time out periods between multi-PC stages, the Cycle Counter will be stepped even though an error occurred. For example, if an instruction pair is B 400 H 401 and a HSB O-E Error occurs on the B 400, the operator would see:

HSB O-E Error Neon --- On
Time Out Neon --- On
Stall Neon --- On
Program Counter --- 0000 (PC-1)
Cycle Counter --- 11 (δ)
Static Register --- H 401

The operator is only able to infer from these indications that there was an HSB O-E Error on the previous γ instruction, and not on the current H instruction.

In order to determine what the instruction was and whether or not it can be corrected, it is necessary to be able to recall the previous instruction. The ability to recall an instruction depends upon the fact that the CC reading is always one greater than the address of the instruction being executed unless a transfer of control (Q m, T m, or U m) was effected in the preceding instruction. In this case the CC reading was altered by an unknown amount, and in the general case the instruction cannot be recalled.

Assume the instruction sequence was

Address	Instruction pair
100	A 900 C 803
101	B 400 H 401

and that the computer stalls in the error situation outlined above. The operator can recall the γ instruction by the following procedure:

Error Correction Procedure A

1. Place IOS in the ONE INSTRUCTION position.
2. Depress Stop Switch to set the Stop FF.
3. Carefully note the error neons lit because operation of the Start Bar will reset the error flip-flops.
4. Clear CY.
5. Push the locking switch at D-3.2 on SC Panel up to the RETAIN C position. This will prevent the addition of one to the CC reading during β.
6. Depress Start Bar. Computer will stop in β time out with the address of the next instruction pair, 102, set up in SR.

7. By use of the SR set up switches, decrease this address by one. This, of course, will not affect the address in CC.
8. Depress the Start Bar once. The computer will stop in γ time out with the left instruction set up in SR. In this example, B 400 would appear.
9. If the instruction can be repeated (see below) depress Start Bar. If an error does not recur, return the switch in step 5 to normal and IOS to CONTINUOUS. Depress Start Bar to resume execution of the program.

The operator can recall the δ instruction with the following procedure:

1. Execute steps 1 through 8 of the above.
2. Replace the γ instruction with a skip order (decimal zeros) by lifting the switch at D.1-18 and returning it to normal.
3. Depress the Start Bar. The computer will stop in δ time out with the right hand instruction set up in SR. In this example H 401 would appear.
4. Execute step 9 of the above procedure.

As noted, these procedures will recall the desired instruction unless one of these instructions involved a transfer of control. Assume that an instruction pair is:

Address	Instruction Pair
100	Q 000 K 400
.	.
.	.
399	C 450 L 300
400	X 000 J 460

and that the contents of rA and rL are equal when executing the Q instruction on line 100. If a HSB O-E Error is detected on the transfer from rA to rL when the computer does the K order, the operator will wish to recall the instruction. However, on the preceding γ the CC reading was changed to 400 because of the Q instruction. Therefore the recall procedure A, will not bring up the contents of memory location 100, but will bring up the contents of 399 instead. The same problem exists with the U and T instructions, and of course may occur on δ as well as γ. The best course to follow when recalling an instruction is to check with the programmer to be certain that the address which sets up in SR on β time (before reducing by one) cannot be reached by a transfer of control operation. In the above example, 400 would set up in SR in β time out. The programmer would recognize that this address could be reached as a result of executing line 399 or transferring from line 100. Unless he can determine which is the previous line by examining the contents of the registers, etc, the only safe procedure is to initiate the general rerun procedure that accompanies the program.

SECTION 3

Central Computer Instructions, Error Analysis

The previous section outlined a procedure for repeating an instruction that had been executed with an error. It remains to determine what instructions can be repeated. In order to facilitate explanation, the instructions are divided into three categories:

a) Single and multiple word transfers.
b) Arithmetic operations including the shifts.
c) Transfer of control operations.

Single And Multiple Word Transfers

Memory-to-register transfers: The instructions B m, L m, F m, V m, Y m may be repeated since none of the operands involved are destroyed by the transfer. Of course the contents of the recipient register are destroyed but this is intended. The E m instruction, however, cannot in general be repeated, since the contents of rA play a part in the transfer. The type of error most commonly detected on memory to register transfers is the presence of an even pulse count in some digit, resulting in an HSB O-E error. Because these transfers are either single Program Counter stage instructions or multiple stage with no time out period between successive stages, the computer will always stall in the time out period of the succeeding stage of the Cycle Counter.

Register-to-memory transfers: If the error was an HSB O-E, which is most common, the H m, G m, J m, W m, Z m, R m instruction may be repeated since the contents of the registers are not altered by the transfer. The C m and 30 m or 40 m instructions cannot be repeated since these orders clear the registers upon read-out.

Register-to-register transfers: The K m instruction, the only instruction in this category, cannot be repeated since it clears rA upon transfer. The α operation is the transfer of the contents of the Control Counter to SR on α time. Since CC is not cleared upon transfer the α operation can be repeated. It is necessary only to clear CY by operating the Clear CY Switch and operate the Start Bar.

Arithmetic Operations

Addition: The A m and S m instructions are two Program Counter stage instructions with a time out period between each stage. On PC-1 the contents of the indicated memory location, m, are transferred to rX. On PC-2 the contents of rA and rX are sent to the Adder and the algebraic sum returned to rA. Thus, if an error occurs on PC-1 the instruction may be repeated since it is a memory-to-register transfer, but PC-2 is not repeatable since one of the operands, the contents of rA, is destroyed.

For example, if an even pulse count is detected on the transfer to rX during PC-1 the computer will stall in PC-2 time out. Thus if the instruction pair is B 700 A 402 the operator will see:

HSB O-E Error Neon --- On
Time Out Neons --- On
Stall Neon --- On
Program Counter --- 0001 (PC-2)
Cycle Counter --- 11 (δ)
Static Register --- A 402

The instruction can be restarted as follows:

1. Place IOS on ONE INSTRUCTION.
2. Depress Stop Switch to set Stop FF's.
3. Clear PC by depressing Clear PC Switch, F-8.2.
4. Depress Start Bar.

If the error does not recur, return IOS to CONTINUOUS and depress Start Bar. As before, if the error persits when repeating steps 3 and 4, a component failure is indicated and an engineer should be notified.

The operations performed on PC-2 are the following: The contents of rA and rX are read into the Adder, and the contents of $\overline{rX}$ and $\overline{rA}$ are read into the barred Adder. The inputs to the Adder from the duplicated X registers are compared for identity and odd-evenness, while the inputs to the Adder from the duplicated A registers are compared for identity and odd-evenness. The sum returning to rA and $\overline{rA}$ from both Adders is checked for identity.

Another checker associated with the Adder is the Adder Alphabetic Error Circuit. This duplicated checking circuit detects the programming error of adding two alphabetic characters in any digit position within a word or of adding two non-sign characters (the sign characters are 0 and -) in the sign position. The Adder Alph Neons, A.2 and A.3-7.2, are lit whenever this condition is obtained.

If one or more of these checks is not passed during PC-2 of an A m or S m instruction, the computer will stall in the next time out period which in this case is in the next CY stage. As an example, if the instruction pair is B 700 A 402 and an even pulse count is detected in some digit of the rX input to the Adder during PC-2, the operator will see:

Adder Subtrahend Error Neon --- On
Time Out Neon --- On
Stall Neon --- On
Cycle Counter --- 00 (α)
Program Counter --- 0000 (PC-1)
Static Register --- binary zeros (no neons lit)

The X m instruction differs from A m and S m in that it assumes the operands are in rA and rX. There is thus no transfer from memory to rX, hence, it takes only one Program Counter stage. PC-1 of this instruction is then the same as PC-2 of A m and S m, and therefore cannot be repeated.

The operations performed on β time include the addition of one to CC. If an Adder error occurs on this operation, the β operation cannot be repeated, since the contents of CC are destroyed.

Multiplication and division: The M m, N m, P m, and D m instructions assume that one operand is contained in rL as a result of a previous order. The second operand is transfered from memory location m into rX by the multiplication or division instruction itself. Thus if the computer stalls during the execution of a multiply or divide order because of an error made during any step of the instruction except the last PC stage the instruction may be repeated by simply clearing the Program Counter by operation of the Clear PC Switch. If the error is made on the last PC step, the computer will stall during the time out of the next Cycle Counter stage. In this case the instruction may be repeated by use of Procedure A.

Shifts: The operand for the shift instructions is the contents of rA. Since this operand is assumed to be in rA at the start of the shift instructions, and shifting itself alters the contents of rA, the instruction cannot be repeated.

Transfer Of Control Operations

Unconditional transfers: The U m instruction is the only instruction in this class. As pointed out in Chapter III, it is accomplished by replacing digit positions 10, 11, and 12 of the word in CC with the information in those same numbered digit positions of CR. An error in executing this instruction is not in general, repeatable, since the CC reading may be destroyed preventing the recall of the U m instruction.

Conditional transfers: These instructions are the Q m and T m. During the first PC stage the contents of rA and rL are sent to the Comparator for the appropriate comparison. A HSB O-E or Comparator error occurring during PC-1 does not prohibit repeating these instructions since none of the operands are destroyed. Clear PC by use of the Clear PC Switch to repeat. However if the error occurs during PC-2 the computer will stall during time out of the next CY stage. The instruction in general cannot then be repeated since the CC reading may have been altered by an unknown amount and thus the instruction cannot be recalled.

SECTION 4

Register Comparator Errors

The four registers which are used to store operands for the arithmetic operations, rA, rL, rF and rX are duplicated. The contents of the duplicate register, for example, rA and $\overline{rA}$ are being constantly compared by means of a half adder as shown in Fig. 1.

As long as the inputs to the half adder are identical there will be no output. However, if the registers differ in even a single pulse position, there will be a sum output from the half adder which will set the error flip-flop. The set output of the error flip-flop will light an error neon in the SC Panel and prevent FF TO from resetting, thus stalling the computer. It should be noted that these register comparisons depend on no function table signals and hence, are independent of the instructions the computer is executing.

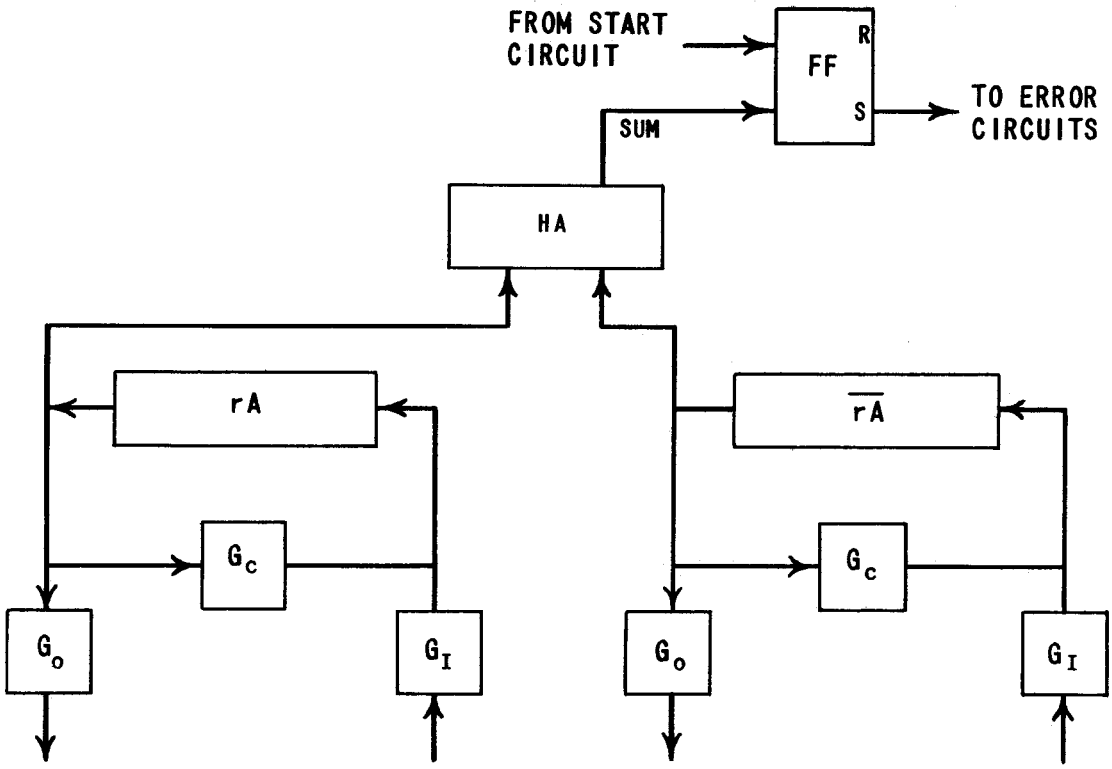


FIG. I

The checking circuit comparing the duplicate A registers controls the neon on the SC Panel labeled "A Comp". Likewise, the corresponding circuits for the L, X and F registers control neons on the control panel labeled "L Comp", "X Comp" and "F Comp".

In the event of a register comparator error, the procedure to recall the last instruction executed must be modified slightly. Operation of the start circuit will reset the error flip-flops. However, FF TO is not reset until a number of minor cycles after the resetting of the error flip-flops. But the register comparators are operating continuously and within one minor cycle following resetting of the error flip-flops the error will be detected again and the computer will stall. Hence, it is necessary to lock down the error delete switch located on the SC Panel under the appropriate comparator error neon after executing the first three steps of the procedure outlined in Section 2. Depressing the error delete switch <u>will not</u> reset the error flip-flop nor prevent it from setting but will prevent the error signal from inhibiting the resetting of FF TO, thus allowing the remaining steps of the procedure to be executed.

The instruction which can be repeated in event of register comparator errors are those which read into the register in error. These are listed below

F	Comp:	F, M, N, P
A	Comp:	B, M, N, P, D
L	Comp:	L
X	Comp:	B, M, N, P, D, L

If for a particular register comparator error, the instruction is other than one listed above, the program should be restarted at a point indicated by the programmer. The appropriate error delete switch must be kept depressed until the error is cleared by reading new information into the register either by an instruction or by initiating the "Initial Clear Procedure (Chapter VI). If a register comparator error persists, an engineer should be notified.

SECTION 5

Static Register Errors

An unacceptable pulse combination can set-up in the the Static Register as the result of a programming error or a component failure. For example ⅈ, ⅉ, ⅊ will not be decoded as instructions in the first instruction digit decoding table nor will single zone alphabetic characters be decoded by the second, fourth, fifth or sixth decoding tables. Also, a component failure such as one of the SR Flip-Flops remaining in the set state regardless of the input signals can cause an unacceptable combination to be set-up. The error symptoms for the various digits of the SR are listed below.

First instruction digit:

An unacceptable first instruction digit will cause the computer to stall in a time-on period. Assume that through a programming error control has been transferred to line 400 which contains the following constant which was intended to be used in an editing sub-routine.

400: ⅈ00000 000000

On ⅋ time-out ⅈ0 000 will set-up in the Static Register. FF TO will reset at the end of a minor cycle, but the pulse combination for ignore (1 00 0000) is not decoded and hence, no function table signals are produced. Because no function table signals are produced, the computer will stall and the operator will see:

Time-Out Neons	- Off
Stall Neon	- On
Program Counter	- 000 (PC-1)
Cycle Counter	- 10 (⅋)
Static Register	- ⅈ 000.

The operator can check that this is actually a programming error by the following procedure:

1. Place IOS on ONE OPERATION.
2. Depress Stop Switch. This will set FF TO and the Stop Flip-Flops.
3. Set up a 50 order by use of the Static Register set-up Switches.
4. Print out the Control Counter reading. (In the above example, the reading would be 401 since the control counter was advanced on β)
5. Print out of tne memory, the contents of the location indicated by the Control Counter minus one. (In the above example this would be 401-1 = 400).
6. If the word from the memory discloses an unacceptable digit, a programming error is indicated. However, if the word from the memory

is correct, a component failure is indicated. In the latter case, the instructions should be recalled. If after several trials, the instruction still fails to set-up correctly an engineer should be notified. NOTE: In case of any Static Register difficulty, be sure that all of the Static Register set-up Switches are in the neutral position.

Second Instruction Digit

The instructions affected by an unacceptable second instruction digit (other than the input-output instructions which are discussed in a separate section) are the shift and skip orders.

On the shift orders, the second instruction digit determines how many places the contents of rA are to be shifted. The contents of rA shift one place on each Program Counter stage. The shift is ended when the output line of the second digit decoding table, which is selected by the second instruction digit, matches the corresponding output line of the Program Counter decoding table. Thus if a shift left eight instruction (;8 000 or 08 000) is given the eight output line of the second instruction digit decoding table is activated which in turn alerts a gate. The second input to this gate is the eight line from the Program Counter decoding network. When the Program Counter steps to eight, the gate produces on output which picks up the shift stop line. This line in turn picks up function table signals which will gate an ending pulse at the end of PC-8 to end the instruction. In this example the contents of rA would be shifted eight placed to the left since the contents of the register are shifted one digit for each Program Counter stage.

However, if a shift instruction is set-up in SR with an unacceptable second instruction digit, the instruction will not end correctly. Because of the unacceptable second instruction digit, there will be no output from the second instruction digit decoding table and none of the shift stop gates will be alerted. The contents of rA will continue to shift until the Program Counter reading 1101 (PC-14) is detected. This will stop the computer and also set the Function Table Intermediate Error Flip-Flop which lights the corresponding neon located at A.3-14 on the SC Panel.

Assume a .4 000 was programmed on γ time but that one of the second digit flip-flops failed to set. The computer would stall with the following indications:

Function Table Intermediate Error Neon	- On
Time-Out Neon	- On
Stop Neon	- On
Program Counter	- 1101 (PC-14)
Cycle Counter	- 10 (γ)
Static Register	- .? 000

The operator can follow the procedure outlined above to determine if the error is due to the programmer. (Remembering to clear the Program Counter). However, the instruction cannot be repeated since the contents of rA have been destroyed.

The skip instruction (00 m) is differentiated from the shift left without sign (on m) by the presence of a zero in the second instruction digit. Therefore, if a skip order is intended, but the second instruction digit fails to set-up properly, the contents of rA will be shifted. In such a case, the instruction obviously cannot be repeated, but the program must be restarted at a point indicated by the programmer.

Fourth Instruction and Fifth Instruction Digits

The fourth and fifth instruction digits together select the desired channel on any instruction involving the memory. There is a checking circuit on the output of both the fourth and fifth instruction decoding function tables to assure that one and only one output line is selected in each table. These checkers control the two error neons located at A.2 and A.3-14.2 on the SC Panel labeled "Tank Selection". The upper neon is controlled by the fourth instruction digit checker and the lower by the fifth instruction digit checker.

When a Tank Selection Error occurs due to an unacceptable fourth or fifth instruction digit or a component failure, the computer will stall in the next time-out period. For example suppose that a B 3⁄0 C 700 is programmed in memory location 400. When the B 3⁄0 is executed, the computer will stall with the following indication*.

Upper Tank Selection Neon	- Off
Lower Tank Selection Neon	- On
Time-Out Neons	- On
Stall Neon	- On
Program Counter	- 0000 (PC-1)
Cycle Counter	- 11 (S)
Static Register	- CO 700

The Tank Selection Error Flip-Flops are not reset by operating the start circuit. Once a Tank Selection Error has been detected, the error flip-flops are not reset until an instruction involving time selection has been executed without error. Hence, to be able to carry out any diagnostic procedures, the tank selection delete switch located on the SC Panel beneath the error neons must be depressed. When the error flip-flops reset, the switch should be returned to its normal position.

When a Tank Selection Error occurs, the operator should determine as outlined above whether or not a programming error is the cause. If the programmed instruction is correct, the instruction should be repeated several times to determine whether or not a permanent component failure exists. If the error proves to be intermittent, the program should be restarted at a point indicated by the programmer.

Sixth Instruction Digit

An unacceptable sixth instruction digit will cause the computer to stall on all instructions which involve a one or two word transfers to or from the memory.

* An HSB O-E error may also show, dependent upon the exact nature of the failure.

The time selection circuits which compare the Time Selection Counter reading and the sixth instruction digit for equality are duplicated. When equality is found, the duplicate circuits set the duplicate Time Selection Flip-Flop which control the neons located at F and F.1-6.1 on the SC Panel. Only one of the time selection circuits checks the sixth instruction digits for odd-even errors. This is sufficient, however, since both TS Flip-Flops must be set in order to transfer information to or from the memory.

If a sixth instruction digit sets up with a correct numeric portion (0-9) but an incorrect check pulse, the unchecked time selection circuit will set its Time Selection Flip-Flop, but the checked circuit will detect the error. Thus, if an A is set up in the sixth instruction digit, the upper TS Neon will be lit but the lower neon will be off. The computer will stall in time-on since all instructions involving one or two word transfers to or from the memory require TS to produce an ending pulse.

Assume that B 47A A 900 is programmed in memory location 900. The computer will stall with the following indication:

Upper TS Neon	- On
Lower TS Neon	- Off
Time-Out Neons	- Off
Stall Neon	- On
Program Counts	- 0000 (PC-1)
Cycle Counter	- 10 (ɣ)
Static Register	- B 47A

If the numeric portion of the sixth instruction digit lies outside the numeric range of the Time Selection Counter, (0-9), the indication will be the same except neither TS Neon will be lit. Thus if a V 74ø sets up in ɣ time in the sixth instruction the computer will stall with the following indications:

Time Selection Neons	- Off
Time-Out Neons	- Off
Stall Neon	- On
Program Counter	- 0000 (PC-1)
Cycle Counter	- 10 (ɣ)
Static Register	- V 74ø

The procedure outlined on Page 114 should be used to determine if a programming error is the cause. However, one modification is necessary, after step 2, (setting the Stop Flip-Flops) the General Clear Switch located at D-2.2 should be depressed to reset the unchecked TS Flip-Flop as it has set.

If the error was due to an intermittent component failure or incorrect programming, the problem should be restarted at a point indicated by the programmer.

SECTION 6

Periodic Memory Check

In continuous operation, the computer is interrupted each three seconds for the Periodic Memory Check. During this check the contents of each ten word

memory channel is read into the HSB O-E Checker. This operation takes place on time and holds up the normal $\propto$ cycle until completed. The PMC Neon at A.3-5.3, when on, indicates that a memory check will be performed during the next $\propto$ cycle. It does <u>not</u> indicate an error.

The check operation consists of three Program Counter stages. PC-1 loads the address of the initial channel (000) into CR and SR. During PC-2, channel 000 is read to the HSB O-E Checker. During this transfer, the contents of CR is sent to the Adder and advanced by ten. At the end of the period required to check the first ten words of the memory, a time-out period is initiated and the next address 010 is set up in CR. This procedure is carried out (on PC-2) until all 100 channels have been checked. At the conclusion of the checking period required for the last channel, (990), the Program Counter is stepped to PC-3. On PC-3, the PMC circuits are de-activated and the normal $\propto$ signals restored. The PMC Neon on the SC Panel is turned off at the end of the cycle.

When an even pulse count is detected in the checking of any channel, the computer will stall on the time-out period which follows the checking of each channel. However, in these time-out periods the address of the next channel to be checked is set-up in SR, hence, the address the operator sees is ten greater than the address of the channel which caused the error.

For example, if an error is detected in channel eleven (address 110), he will see:

HSB O-E Neon	- On
Time-Out Neons	- On
Stall Neon	- On
PMC Neon	- On
Program Counter	- 0001 (PC-2)
Cycle Counter	- 00 ($\propto$)
Static Register	- 00 120

These error indications will hold for all except the last channel to be checked, (address 990). The computer is stepped to PC-3 at the conclusion of the checking period for this channel. Hence, in case of an odd-even error the computer will stall with the following indications:

HSB O-E Neons	- On
Time-Out Neons	- On
Stall Neon	- On
PMC Neon	- On
Program Counter	- 0010 (PC-3)
Cycle Counter	- 00 ($\propto$)
Static Register	- 00 000

The procedure for the operator to follow in event of a HSB O-E error on PMC is as follows:

1. Place the IOS in the ONE INSTRUCTION position.
2. Carefully note the address of the channel in error.
3. Depress the Start Bar.

It is possible that more than one channel may be in error. Note the address of all channels that do not pass the odd-even check and repeat step 3 until the computer stops in β time-out.

An HSB O-E error indication on PMC, tells the operator that one or more words within a channel contain incorrect digits. It is often desirable to determine which words are incorrect since the programmer may be able to supply the proper words to be typed in.

Following step 3 of the above procedure the operator should continue:

4. Depress the Start Bar. The Computer will stop on γ time-out.
5. Lock the switch located at D-3.2 in the RETAIN INSTRUCTION position. This will prevent the Cycle Counter from advancing.
6. (a) By use of the Static Register Switches, set-up a 50 m instruction, where m is the address of the incorrect channel.
   (b) Lift the switch at D-2.2 to clear FFTS (See note at end of section).
7. Place the IOS on ONE OPERATION.
8. Clear PC.
9. Depress Start Bar twice. The contents of the selected memory location will be typed in the SCP if the word is correct. If the word is incorrect, the HSB O-E Neon will be lit since the transfer from memory to the output synchronizer is checked. The printer will stall on the first digit that has an even pulse combination. The incorrect digits can be by-passed by manually operating the space bar on the SCP.

   For example, assume the word to be printed out of the memory is B00400 H400 900 and the 4 has picked up a pulse giving it an even pulse combination. When step 9 has been executed, the HSB O-E Neon will come on. The SCP will print B00 and then stall. (The incorrect pulse combination will be set-up in the printer neons). Operate the space bar and 00H00 900 will type out. Assuming the printer function switch set to Computer Digit the printed word will be:

   B00 00H00900x

   The operator should note the address of the memory location if the word is incorrect.
10. By use of the Static Register Switches increase the address of the 50 m order by one.

    Repeat steps 8, 9 and 10 until all ten words of the channel which failed to pass the PMC have been printed out. Check other channels if more than one failed in PMC.
11. Clear PC.
12. If the programmer wishes to type in corrections follow the procedure outlined in Chapter 6, section 6. If not, return the "Retain Instruction" switch to the neutral position and restart the program at a point indicated by the programmer.

    If errors on PMC persist, an engineer should be called.

In certain circumstances errors are to be expected on PMC and the above procedure should not be used. For example assume there has been an HSB O-E error during the execution of H 405 on γ. This means an incorrect word has been delivered to the memory. In the procedure to recall the instruction, it is necessary to execute the α cycle. If a PMC is performed at this time an error certainly will be detected when channel 400 is checked.

However, since the order is to be repeated it is not necessary to go through the procedure outlines above unless additional channels prove incorrect.

NOTE: On non-continuous operation, the Y and Z instructions cause the Time Selection Flip-Flops to set during the time out period preceding the execution of these orders. If a new instruction involving TS, such as the 50 m, is to be inserted FFTS must be reset manually after the new instruction is set-up. If this is not done and the Start Circuit is operated the instruction will be executed without regard to the TS Counter since FFTS is already set. The Start Circuit always operates at a specific time within a major cycle. (Major Cycle = 10 minor cycles). The timing is such that the zero word of the specified channel will be read out if the Time Selection Flip-Flop has not been manually reset. Thus with FFTS set and a B405 inserted depressing the Start Bar will cause the contents of memory location 400 not 405 to be read into register A.

SECTION 7

Tape Input Error Diagnosis

Information is read from the tape digit by digit. As each digit is sensed, it is transferred to one of two one-word registers called the Input Synchronizer (SYI_1 and SYI_2). Digits are transferred to one SYI until twelve (one word) have been accumulated. The next twelve digits will be accumulated in the second SYI. During this time the word accumulated in the first SYI will be transferred to the 60 word input storage, called the I register (rI). When the second SYI contains twelve digits, their functions are reversed once more, and the SYI used initially once more begins to accumulate digits from the tape while the second SYI transfers its word to rI. The SYI's are then used alternately until sixty words (one block) have been read from the tape.

As each digit read from tape is transferred to an SYI, it is checked to ensure that it contains an odd number of pulses. The checking circuit is duplicated and controls the two neons located at A.2-15.3 and A.3-15.3 on the SC Panel and labeled IS O-E. (Input Synchronizer Odd-Even).

When an odd-even error is detected on a read operation, the computer does <u>not</u> stall on the next time-out. It <u>will</u> stall on the next read order or on a write or rewind order calling for the servo on which the error occurred. For example assume the following sequences of operations:

a).
```
100: B 300
                31 940
101: H 400
                82 000
102: C 920
                33 760
```

b).
```
100: B 300
                31 940
101: H 400
                81 000
102: C 920
                33 760
```

In both sets of instructions assume that an IS O-E error is detected on the 31 940 order. In sequence a), the computer will stall on the next read order 33 760. In sequence b), however, the computer will stall on the rewind order in line 101 since it calls for the servo on which the read error was made.

In sequence a), the computer will stall with the following indication:

IS O-E Error Neons	- On
Time-Out Neons	- Off
Stall Neon	- On
Program Counter	- 0000 (PC-1)
Cycle Counter	- 11 (8)
Static Register	- 33 760

In sequence b), the indication would be the same with the exception of the Static Register which would contain 81 000.

In general, the operator cannot tell on which servo a read error has occurred by examining the selector digit (second instruction digit) of the Static Register. He can tell however, by examining the read lights located on the Uniservos. In the example listed above, the operator would see the read light and forward light on Uniservo #1 turned on.

The general procedures in case of a read error is to repeat the operation. It is not necessary to recall the instruction that initiated the erroneous read, since the SC error neons and the Uniservo neons give the operator sufficient information to set-up the necessary instruction by means of the Static Register Set-up Switches.

The following procedure will repeat the read operation.

1. Place IOS on ONE INSTRUCTION position.
2. Depress the Stop Switch.
3. Depress the switch at D-3.2 of the SC Panel to the "Retain Instruction" position.
4. Carefully note on which servo the error occurred. (Operating the IS Error Clear Switch will extinguish the read lamp on the servo.)
5. By means of the Static Register Set-up Switches, set up a 1n 000 or 2n 000 order. Do not lock the switches. Choose the order which will reverse the direction of motion of the tape. That is, if the servo lamp indicates that the tape was moving forward when the error occurred, then a 2n 000 is used. If the tape was moving backward, then a 1n 000 order is used. Of course, n is the number of the servo on which the error occurred.
6. Depress the Input Synchronizer Error Clear Switch located at B.3-17.2 of the SC Panel. This will reset the read error flip-flops and extinguish the read lamp on the servo. NOTE: The read error flip-flops are not reset by operating the Start Bar.
7. Depress the "Clear rI and rO" switch located atF-15.1 of the SC Panel. This will clear the erroneous information from rI.
8. Depress the Start Bar. The read will be executed and may read correctly. However, the tape must be repositioned.
9. Clear rI.
10. Set-up a read order which will move the tape in its original direction of motion. That is in the same direction that it was moving when the read error occurred.
11. Depress the Start Bar.
12. If the read is executed without error, return the "Retain Instruction" switch to its neutral position. Place IOS on continuous. Depress the Start Bar.

The Gain Control Switch located at B.2-5.3 of the SC Panel adjusts the sensitivity of the reading mechanism. If the error recurs on step 11 of the initial reread, steps 5-11 should be repeated with the gain switch first in the upper position and then with the gain switchin the lower position.

The case sometimes arrives in repeating a read operation, that the tape will read correctly in the direction opposite to that in which it was moving when the error occurred, but will not read correctly in the original direction. For example, assume that an IS 0-E occurred in executing a forward read on servo 1. In carrying out the reread procedure, the block reads correctly on the 21 000 instruction, but the error occurs consistently on the 11 000 order.

Of course, if the read on step 8 of the reread procedure is executed without error, the information is exactly what is desired. However, the tape will be incorrectly positioned and repositioning the tape on step 11 will refill rI with incorrectly read information. Under these conditions it is desirable to

transfer the information from rI to the memory after step 8. But to do this it is necessary to know the address of the memory location designated to receive this information. There are two situations in which this address will be known without having an intimate knowledge of the program. One, the computer stalls after a IS O-E error on a 3n m or 4n m instruction. Or the computer stalls on a 30 m or 40 m instruction.

To illustrate, assume the following sequence of instruction:

(a)		(b)		(c)	
100: B 400	31 940	100: B 400	31 940	100: B 400	31 940
101: H 702	53 460	101: H 702	82 000	101: H 702	51 240
102: A 453	41 760	102: A 453	30 760	102: A 453	39 760

Assume that a IS O-E error in each case in executing the 31 940 order. In sequence a), the computer will stall on the 41 760 in line 102. The 760 is the initial address of the sixty memory locations which are to receive the information transferred to rI on the previous read. Likewise, in sequence b), the computer will stall on the 30 760 instruction. Again the 760 indicates the address to which the contents of rI are to be transferred. However, in sequence c), the computer will stall on the 51 240 order in line 101. It is not a read order, but it does involve the servo on which the read error occurred. In such a case, the operator does not know the memory location to which the contents of rI are to be transferred. Hence, the following procedures can be used only if the computer stalls after an IS O-E on an order which involves the transfer of the contents of rI to the memory.

Before using either of the following procedures, the operator should try the normal reread procedure on all settings of the gain switch. If the block reads correctly in step 8 but not in step 11 the following procedure may be used if the computer stalled initially on a 3n (m) or 4n (m) instruction.

1. Execute steps 1 through 8 of the normal reread procedure.
2. With the "Retain Instruction" Switch depressed, the instruction on which the computer initially stalled will set-up in the Static Register each time the Start Bar is depressed. In this case it will be either a 3n m or 4n m instruction. By use of the Static Register Set-up Switches, change the second instruction digit to a zero (10011), thus changing the instruction to a 30 m or 40 m order. All five switches of the second instruction digit must be locked in the one (down) or zero (up) position.
3. Depress the Start Bar. This will transfer the contents of rI to the memory.
4. Unlock the second instruction digits Static Register Set-up Switches.
5. Execute steps 9 through 11 of the normal reread procedure. This will reposition the tape.
6. Depress the IS Error Clear Switch.
7. Depress the "Clear rI and rO" Switch.
8. By means of the Static Register Set-up Switches, change the first

instruction digit to 1 (000 0100) if it is a 3; or to a 2 (000 0101) if it is a 4.

9. Return the "Retain Instruction" Switch to its neutral position. Place IOS on CONTINUOUS.
10. Depress the Start Bar.

The following procedure may be used if the computer stalls on a 30 m or 40 m instruction:

1. Execute steps 1 through 8 of the normal reread procedure.
2. The 30 m or 40 m order on which the computer originally stalled will set-up in the Static Register.
3. Depress Start Bar. This will transfer the contents of rI to the memory.
4. Execute steps 9 through 11 of the normal reread procedure. This will reposition the tape.
5. Depress the IS Error Clear Switch.
6. Depress the "Clear rI and rO" Switch.
7. By means of a Static Register Set-up Switches change the first and second instruction digits to zero.
8. Return the "Retain Instruction" Switch to its neutral position. Place IOS on CONTINUOUS. Depress the Start Bar.

All of the procedures listed above may be used if a Tape Check Error occurs with or without an accompaning IS O-E Error. A Tape Check Error is indicated by the neon located at A.2-16.3 of the SC Panel.

If the block cannot be read correctly in either direction, the operator can try moving the Milar spacer* on the servo and repeating the reread procedures. If the block still cannot be read, the program should be restarted a point indicated by the programmer.

*The Milar spacer is a thin plastic film that lies between the read-write head on the magnetic tape. It is used to reduce friction and wear on the head and tape.

CHART A

ARITHMETIC AND MEMORY CIRCUITS OF AN ELEMENTARY COMPUTER

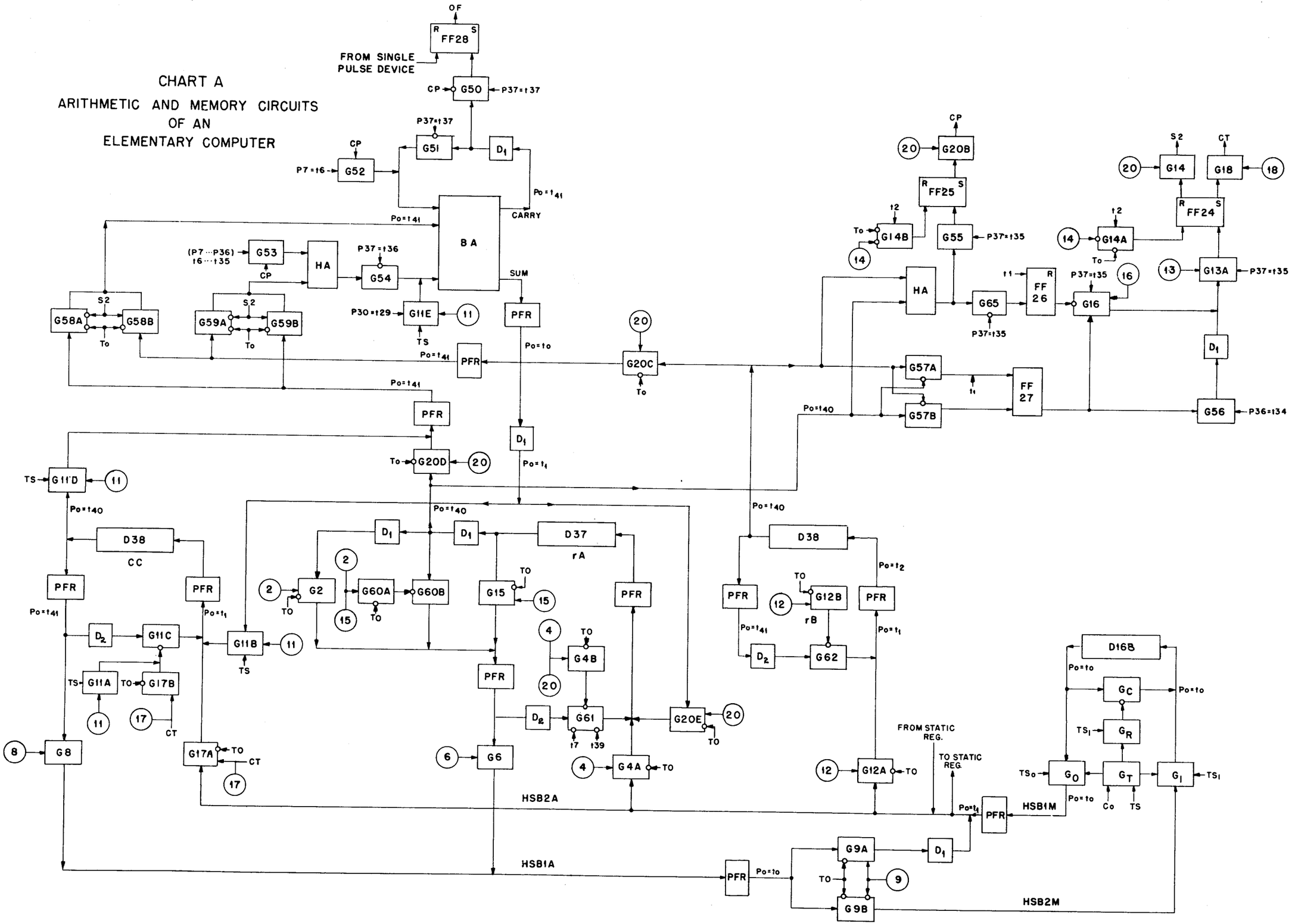
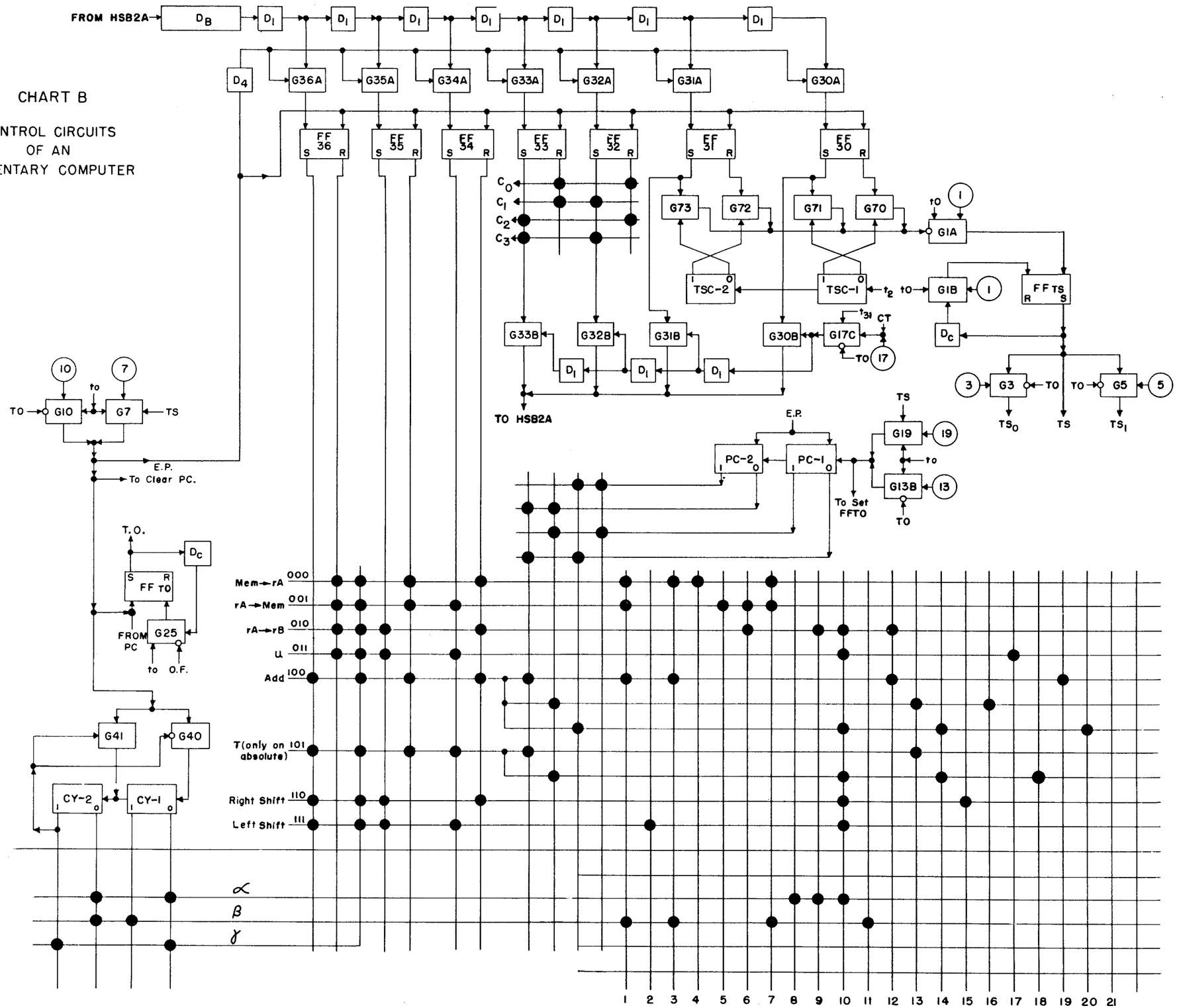


CHART B

CONTROL CIRCUITS
OF AN
ELEMENTARY COMPUTER



XS3 / ZONE	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1	Δ	—	0	1	2	3	4	5	6	7	8	9	'	&	(
01	r	,	.	;	A	B	C	D	E	F	G	H	I	#	¢	@
10	t	"	|	) /	J	K	L	M	N	O	P	Q	R	$ ¥	*	? ø
11	Σ	β	:	+	/ +	S	T	U	V	W	X	Y	Z	% ø	=	

■ = NEW PULSE COMBINATION

[A / B] = A - NEW SYMBOL REPRESENTATION B - OLD SYMBOL REPRESENTATION

Above is the 63 character UNIVAC excess-three binary pulse code. The expanded code, using 63 of the possible 64 pulse combinations consistent with UNIVAC logic, has been a development included in the design of the new auxiliary equipment. The Unityper II and the High Speed Printer make use of the expanded code.

CHART C

MANUAL CONTROL
TO ALL UNITS
MONITORING SIGNALS
FROM ALL UNITS

SUPERVISORY CONTROL (SC)

CONTROL BUSSES
READ BUS
TO AND FROM ALL UNISERVOS
WRITE BUS

STANDARD PULSES TO ALL UNITS

TIMING PULSE GENERATOR & CYCLING UNIT (CU)

CONTROL SIGNALS TO GATES

FUNCTION TABLE (FT)

CY

PC

TIME OUT (TO) SIGNALS TO GATES

MAIN MEMORY (MM)
(1000 WORDS)

MEMORY SWITCH

CONTROL SIGNALS

HSB 1M
HSB 2M

INPUT-OUTPUT CONTROL & INTERLOCK CIRCUITS

CHECK CIRCUITS

CONTROL CIRCUITS

INSTRUCTION DIGITS

MEM. LOCATION DIGITS

STATIC REGISTER (SR)

DISTRIBUTOR LINE

CONTROL SIGNALS TO INPUT-OUTPUT CONTROL & SYNCHRONIZERS

INPUT SYNCHRONIZER
I
(60 WORDS)

OUTPUT SYNCHRONIZER
O
(60 WORDS)

INPUT FROM REGISTERS AND OTHER UNITS

MQ 13 SIGNAL TO CR

CONTROL SIGNALS TO A, F, L, & X REGISTERS TO AA & CP AND TO CONTROL CIRCUITS

CONTROL SIGNALS TO AA & CC

OVERFLOW SIGNAL TO MQC

ALGEBRAIC ADDER (AA)

MQC

COMPARATOR (CP)

SIGNAL < 3

SIGNAL ≥ 3

ODD-EVEN CHECKER (OEC)

HIGH SPEED BUS AMPLIFIER (HSBA)

HSB 1I

HSB 2O

SIGNALS TO CONTROL CIRCUITS

Y (10 WORDS)

V (2 WORDS)

L (1 WORD)

F (1 WORD)

X (1 WORD)

A (1 WORD)

CC (1 WORD)

CR (1 WORD)

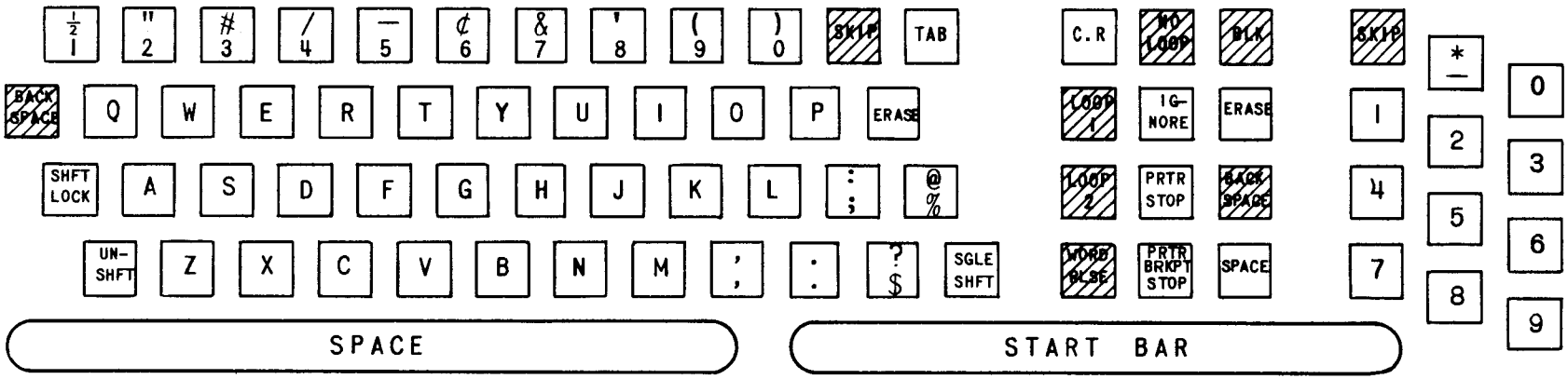
HSB 2A

HSB 1A

EXTRACT CONTROL

LEGEND:
———INFORMATION SIGNALS
- - - - - -CONTROL SIGNALS & PULSES

SIMPLIFIED BLOCK DIAGRAM OF THE UNIVAC CHART D



SUPERVISORY CONTROL KEYBOARD

*KEY LABELED "WORD RELEASE" FOR SUPERVISORY CONTROL IS USED ON UNITYPER AS LOOP 3

ADDITIONAL KEYS USED ONLY ON UNITYPER KEYBOARD

CHART E

HEATERS ON
HEATERS OFF
RESTORE
OFF
STAND-BY POWER
D.C. ON
D.C. OFF
HIGH
NORMAL
I.F. BIAS CONTROL

INITIAL CLEAR 1 — CU-TSC CLEAR
INITIAL CLEAR 2 — CU START INTERLOCK
CLEAR FF TS — CU START
INHIBIT PMC — GENERAL CLEAR
RETAIN C — MEMORY CLEAR
GROUP 1 — RETAIN INSTRUCTION
O_p O — GROUP 2 OSCILLOSCOPE
pO — TSP SYNC

REGISTER SELECTOR: HSB, F F, L L, A A, X X, V V, CR, M, O I, TC CU, TPG, C

MEMORY SELECTOR - HUNDREDS: 00 10 20 30 40 50 60 70 80 90

MEMORY SELECTOR - TENS: 0 1 2 3 4 5 6 7 8 9

SECONDARY SELECTOR: 7p 0, 7p 1, 13p 2, 13p 3, 27p 4, 5, SYI 6, SYO 7, 8, 9

BAY SELECTOR: BAY A, BAY B, BAY C, BAY D, BAY E, BAY &, BAY N, BAY G, BAY H, BAY J

D.C. FAULT INDICATORS: BAY K, BAY L, BAY P, DE CORNER, POWER SUPPLY, UNI SERVOS, P, P, T

FAULT TEST

PRIMARY FAULT RELAY RESET

STAND-BY HEATERS - LONG TANKS: F, L, A, X, C, CR1, SYI1, SYO1

UNISERVO SCREEN FUSE

AGC

STAND-BY POWER (HEATER)

TIME-OUT RESET

HSB STALL SPEAKER

HEATERS - SHORT TANKS: F, L, A, X, V, CR2, SYI2, SYO2

PMC

HIGH

A OFF S

HSB TO MEM. | HSB TO SYO | LOW GAIN | HSB O-E | HSB COMP. | ADDER MIN. | ADDER SUB. | ADDER ALPH. | ADDER COMP. | F COMP. | L COMP. | A COMP. | X COMP. | COND. TRANS. | DELETE SELECTOR | MASTER DELETE | RESET CONTROL | X÷ ERROR STOP | HSB ERROR INSERT | B C.U. ERROR INSERT | ON SERVO POWER | TIME OUT | CYC UNIT | F.T. OUTPUT | F.T. INTER. | TANK SEL. | SERVO SEL. | I.S. >720 | I.S. O-E | O.S. O-E | I-O INT. | TAPE CHECK | R I.S. ERROR INSERT | I.S. ERROR CLEAR | O.S. ERROR CLEAR

FIRST INSTRUCTION DIGIT | SECOND INSTRUCTION DIGIT | FOURTH INSTRUCTION DIGIT | FIFTH INSTRUCTION DIGIT | SIXTH INSTRUCTION DIGIT

← DECIMAL ZERO →

← BINARY ZERO → INST. MSR SR

IER | OR | IER-OR | ≧3 | REPEAT | MULT.-QUOT. COUNTER | PROGRAM COUNTER

STALL | STOP | STOP | TO | TS | S1-C^p | S1-X | CLEAR CY | CYCLE COUNTER | CLEAR PC AND $\overline{PC}$ | CLEAR $\overline{PC}$

CONDITIONAL TRANSFER BREAKPOINT SELECTOR: ALL, 0 1 2 3 4 5 6 7 8 9

CONDITIONAL TRANSFER | TRANSFER: NO TRANSFER, BREAK POINT

TANK HEATER VOLTAGE MONITOR

VOLUME

ONE OPERATION | ONE STEP | ONE ADDITION | ONE INSTRUCTION

OVER FLOW | 0 1 2 3 4 5 6 7 8 9 – | SECOND INSTRUCTION DIGIT

INITIAL READ | IR TIMER | CLEAR C | SCI CR

–	0	1	2	3	4	5	6	7	8	9
,	.	;	A	B	C	D	E	F	G	H	I
SCI CR	J	K	L	M	N	O	P	Q	R	
S	T	U	V	W	X	Y	Z			

BLOCK SUBDIVISION SELECTOR: 1 2 3 4 5 6 7 8 9 –

INPUT SYNCHRONIZER COUNTER | INPUT TANK COUNTER | CLEAR I AND O | OUTPUT SYNCHRONIZER COUNTER | OUTPUT TANK COUNTER | DIRECTION STORAGE | REVERSE FIRST BLOCK | READ INT. | WRITE INT. | REWIND OVERLOAD

INITIAL TAPE SELECTOR: 1 2 3 4 5 6 7 8 9 –

OUTPUT SELECTOR: M, A, X, L, F, C, CR, SYI, EMPTY

SCI CR FILL | INPUT READY | SYI1 | 12 TH DIGIT INPUT ERROR | SYI2 | OUTPUT READY | SKIP OUTPUT BREAK POINT

ABSOLUTE REF. VOLTAGE

PERCENT OF ALLOWABLE DEVIATION

TANK HEATER VOLTAGE

VOLTAGE MON. REMOTE METER

0	1	2			3	4	5
0	1	2			3	4	5
+750 0	+90 0	+8 -74	0	0	-4 -74	-88 -117	+290 +90
+410 +280	+80 +70	+5 0	1	1	-14 -34	-88 -150	+280 +246
+380 +90	+80 +60	+5 -14	2	2	-20 -40	-95 -100	+246 +165
+375 0	+80 0	0 -175	3	3	-20 -100	-95 -117	+120 +90
+280 0	+80 -106	0 -11	4	4	-25 -100	-100 -175	0 -34
+246 0	+75 0	0 -13	5	5	-25 -190	-100 -216	-34 -74
+200 0	+70 +60	0 -14	6	6	-34 -40	-117 -150	-88 -100
+165 +90	+70 0	0 -15	7	7	-34 -50	-117 -190	0 -191S
+165 +60	+66 +60	0 -16	8	8	-40 -55	-125 -190	-166S -191S
+160 -40	+60 0	0 -17	9	9	-40 -100	-140 -190	-191S -217S
+150 0	+50 0	0 -18	0	10	-50 -150	-150 -163	+48 0
+120 +105	+40 0	0 -20	1	11	-50 -190	-150 -165	
+120 -30	+40 -88	0 -21	2	12	-55 -100	-150 -190	
+105 +95	+35 0	0 -40	3	13	-55 -150	-150 -216	
+105 +80	+30 0	0 -50	4	14	-60 -88	-165 -216	
+95 0	+30 -40	0 -60	5	15	-60 -100	-175 -216	+279 +90
+90 +84	+20 0	0 -100	6	16	-71 -88	-190 -203	+247 +90
+90 +80	+15 -100	0 -106	7	17	-74 -88	-190 -300	+79 0
+90 +70	+8 0	0 -300	8	18	-74 -190	-216 -229	-74 -86
+90 +60	+8 -14	-4 -34	9	19	-80 -150	-216 -300	-100 -150

MEASURE

DELAYED SHUT-OFF

VOLTAGE MON. MANUAL SWITCHING

HIGH

VOLTAGE MON. FAULT RELAY RESET

LOW

EL-170

CHART F