PROGRAMMING MANUAL

for the

ROYAL PRECISION

RPC-4000

Electronic Computing System

ROYAL McBEE CORPORATION

Electronic Data Processing

PREFACE

This manual for the RPC-4000 Electronic Computing System contains the information necessary for programming and operating the basic RPC-4000 computer. The manual has been prepared for both experienced programmers and those with no experience.

Beginning programmers will be particularly interested in the "Introduction to Computers" (Section 1), "Arithmetic for Programmers" (Section 2), and "Programming Techniques" (Section 5).

Section 5 contains an extensively detailed sample program. Even programmers who have considerable knowledge of data processing will find this section valuable. It reveals many aspects of RPC-4000 programming that are best explained by illustration.

The expositions on "The Central Computer" (Section 3), "The Instruction List" (Section 4), and the "Manual Controls and Operating Procedures" (Section 6) present the essential information on the design and use of the RPC-4000. Charts, diagrams, and summaries have been extensively employed so that the manual will have continuing value as a reference work for the RPC-4000 user.

CONTENTS

List of Illustrations

AN INTRODUCTION TO COMPUTERS

1

The emergence of the electronic computer as a major tool for Business and Industry is not surprising when it is considered that a very large part of today's total work effort is devoted to the processing of lengthy calculations or vast amounts of statistical data. Much of this computation involves wearisome repetition along with the necessity to effectively utilize a confusing array of interrelated information. It is just this sort of job that can be handled best by an intelligently directed machine. Not only can the computer perform these tasks many times faster, but with much less fallibility that can the human worker.

Since earliest times, the development of computing aids has followed two separate paths, depending upon the means employed to recognize and to represent information values. Those devices which operate by the measurement of continuous physical variables are known as analog devices or Analog Computers. A fuel gauge is a simple analog device in that the deflection of a pointer is analogous to the quantity of fuel in a tank.

The other type of computing aid, and that which directly concerns us, is known as a Digital Computer. It is characterized by its representation of values, both quantitative and symbolic, by counts of discrete discontinuous physical units. The abacus is a simple digital device in which beads are used to represent the counting units. A more complex and modern, but still non-automatic digital device is represented by the desk calculator.

THE AUTOMATIC DIGITAL COMPUTER

The abacus has served man for thousands of years as a simple static storage device to hold the progressive results of lengthy calculations. The modern desk calculator has, in addition to a similar limited storage capacity, the ability to mechanically perform certain basic arithmetic operations, such as addition, division, etc. However, both of these aids to computation require the constant services of the human operator to direct each individual operation. Thus the speed of these non-automatic devices is limited to the rapidity with which the operator can control their actions. In order to function automatically, a computer must

be self-sequencing; that is, it must have the capability of controlling the order in which the steps of a calculation are performed, by reference to a series of coded signals which are stored within itself.

The history of the Automatic Digital Computer goes back to the year 1822, when an Englishman by the name of Charles Babbage, with the financial backing of the British Government, began work on what he called a full size "Difference Engine". This machine was designed for the purpose of calculating mathematical tables so as to relieve the human worker of this routine function. Ten years later, after an expenditure of 17,000 pounds, the project was abandoned. In 1833 he elaborated upon his initial efforts to develop the concept of a universal computer which he called an "Analytical Engine". It was designed to be fully automatic and externally programmed. It incorporated facilities for input/output, arithmetic operations, internal storage and automatic program control. Unfortunately, the state of the engineering art was insufficiently advanced to produce a machine of such mechanical complexity. Although a considerable amount of money and effort was expended, the machine was never completed.

It was not until the year 1944 that the first such universal digital computer was actually completed. This machine, generally referred to as the Harvard Mark I, used electromagnetic relays and mechanical counters, and was extremely cumbersome compared with the computers in use today. The first computer to substitute electronic circuitry for electromagnetic was the ENIAC (Electronic Numerical Integrator And Calculator), which was used primarily to solve ballistics problems. It contained some 18,000 vacuum tubes and about 1500 electro-mechanical relays.

The growth rate of the computer industry following these pioneer efforts has been truly fantastic. Digital computers are being entrusted with an ever increasing share of the routine, repetitive functions of business and industry. Computers are being produced in a variety of types and sizes depending on their intended use. Of particular importance is the class of small to medium size computers being applied to such diverse problems as process control, data reduction for management analysis, inventory control and scientific problem solving. They are characterized by moderate cost, ease of installation and maintenance, and great flexibility in their application to a variety of tasks.

COMPUTER DESIGN CONSIDERATIONS

An automatic digital computer must, of necessity, contain certain basic logical elements. It must, of course, have the ability to perform a number of simple arithmetic and logical operations. The solution of most mathematical equations, regardless of complexity, can be reduced to a series of basic arithmetic operations. Thus, the ability to add, subtract, multiply and divide is sufficient to perform virtually any mathematical computation. The circuitry which performs the arithmetic

and logical processing is generally referred to as the Arithmetic Element. It receives raw data from the Memory Element and, after acting upon it, returns it to the Memory Element.

In order that information (both instructions and operands) may be supplied to the Arithmetic Element at a rate comparable to its inherent processing speed, it is necessary that there be some form of internal Memory Element. This element is generally used to store both the program to direct the processing operation, and the data to be processed. It must provide rapid access and locatability of each piece of information on a program demand basis. Without the Memory Element, the speed of a computer would be restricted to a mechanical insertion rate.

Prior to beginning a processing operation, and often during the course of an operation, information from an external source must be entered into the computer's memory. Consequently, there is required an Input Element to accomplish this loading of data.

Conversely, at the completion of, and often during an operation, the processed data must be externally presented in a usable and understandable manner. This internal to external transfer of data is a function of the Output Element.

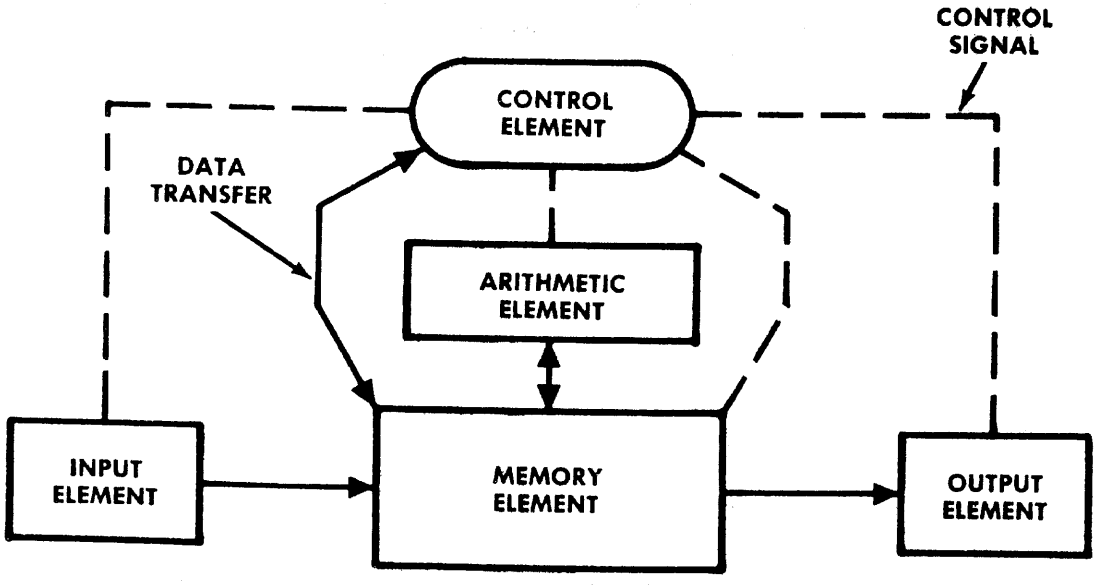


FIGURE 1. Logical Diagram of Automatic Computer

Finally, and vital to the automatic functioning of the computer, there must be a Control Element to coordinate the activities of the other four elements. It is responsible for the proper sequencing of each action within the computer. For most computers this controlled sequencing determines whether a piece of information stored in memory is to be used as an instruction or as data to be processed. Data and instructions may occur intermixed in memory, and in the same form. In fact, the identical piece of information may be used in both ways at different points in a program. Further, this Control Element must activate input and output devices when called for, and, in general, direct the overall performance of a program or program complex.

GLOSSARY OF COMPUTER TERMS

ACCESS TIME----------The time required to bring a selected word from storage to the point at which it is to be used or processed.

ACCUMULATOR----------That register, within the arithmetic element, in which are formed the results of arithmetic and logical operations.

ADDRESS--------------A character, or string of characters, used to identify a location within the computer memory.

ADDRESS--------------A label, usually numeric, which identifies a location
(Absolute) in memory independent of any reference location.

ADDRESS--------------A numeric label which identifies a location in memory
(Relative) relative to some other memory location.

ADDRESS--------------A label, consisting of arbitrarily chosen symbols, to
(Symbolic) represent a location within a program which is independent of the location of the program in memory or its initial location.

ALPHA-NUMERIC--------A symbol system in which are included alphabetic, numeric and special characters.

ARITHMETIC-----------That part of a computer in which arithmetic and logical
ELEMENT computations and decision making functions are performed.

ASSEMBLER------------A utility program which assigns absolute address values for the values in a symbolically addressed program and sets up storage allocations for its various parts.

BASE-----------------In a number system employing positional notation, the base is the number of counts required in each position to cause a change in the next higher position. It is also the number of discrete numeric characters employed in the system.

BINARY CODED---------Representation of each decimal character in a number
DECIMAL by a pattern of binary digits.

BIT------------------Common programming expression for "binary digit". The smallest meaningful unit of information in the computer. An individual bit is restricted to the values "0" and "1".

BLOCK----------------A group of related computer words or characters processed or transferred as a unit.

BOOTSTRAP----------A procedure for entering a program into the computer. The initial few steps of the routine, normally entered manually, are used to automatically load the remainder of the program.

BUFFER----------An intermediate storage device for coordinating the transfer of information from one part of the computer to another.

COMMAND----------The directive portion of an instruction. The specified action to be taken by the computer.

COMPILER----------A utility program which produces a machine or assembly language program from a program which is coded in a problem oriented language. The coding form of the program to be compiled ordinarily will closely approximate standard algebraic notation.

COMPUTER----------An electronic device for the automatic calculation of
(Digital) sequences of arithmetic and logical operations. Quantities and values are represented by patterns of bistable magnetic or electronic indicators.

CONSTANT----------A value which is not subject to change during an operation.

CONTROL ELEMENT------That part of a computer which directs the sequencing and timing of its actions.

DATA----------That information used as operands in the arithmetic and logical operations of the computer.

DATA-REDUCTION-------The processing of large volumes of raw data so as to condense and simplify it to a more meaningful presentation.

DEBUGGING----------The process of eliminating errors from a program by inspection or machine testing.

EXTRACT----------To clear selected portions of a word to zero, leaving the remaining portions intact.

FIELD----------A defined space within a computer word or information format which is assigned to hold a specified type or class of information.

FIXED POINT----------That system of programming arithmetic in which the location of the decimal or binary point in a computer word must be controlled and manipulated by the programmer.

FLOATING POINT-------That system of programming arithmetic in which the location of the decimal or binary point in a computer word is automatically manipulated and controlled by the computer or a program routine.

HEAD-----------------The assembly for recording or reading one track of information on a magnetized surface.

HEXADECIMAL----------The positional number system using a base of 16. A number system which employs 16 discrete numeric characters.

INDEX REGISTER-------A register to contain a quantity which may be used to automatically increment or decrement the address portion of an instruction.

INPUT----------------Information entered into a computer's memory from an external source.

INSTRUCTION----------A set or string of characters which completely specifies one action to be taken by the computer.

LOAD-----------------To enter information into the computer from an external source. Also, to place a value into a register, such as the Index Register.

LOOP-----------------A programming technique in which a sequence of instructions is repeated a specified number of times before proceeding with the remainder of the program.

MAGNETIC DRUM--------A rotating cylindrical drum used for information storage. Recording is in the form of magnetized spots on the surface of the drum.

MEMORY---------------The primary internal storage area of a computer. Generally, the storage device permitting the most rapid access to its data. It is from this storage that instructions and operands are obtained during execution of a program.

MICROSECOND----------One millionth of a second.

MILLISECOND----------One thousandth of a second.

MNEMONIC-------------A code form of identification devised so as to assist in the remembrance of its meaning.

OPERAND--------------An item of information which is to be operated upon, or one which enters into an operation.

OPTIMIZE-------------To code a routine in such ways as to minimize the total memory access time.

OUTPUT----------Information transferred from the computer's memory to an external device.

OVERFLOW----------The generation, in a computer register, of a quantity beyond the capacity of the register.

PARAMETER----------An item which may be assigned arbitrary values, depending upon its use in a given routine.

PARITY BIT----------An extra binary digit added to an item of information for validity checking purposes.

PARITY CHECK----------A method of verifying the accuracy of a data transfer by counting the number of "1" bits in the transferred item, including the parity bit. An accurate transfer is indicated by an even count in an "even parity" system, or by an odd count in an "odd parity" system.

PROGRAM----------A complete sequenced set of computer instructions designed to carry out a desired processing function, or solve a defined problem.

REGISTER----------The hardware for storing one complete computer word.

ROUTINE----------A sequenced set of computer instructions, part of a program, for performing some well defined function.

SECTOR----------The space on a storage drum, measured along the circumference, required to hold one computer word.

SUBROUTINE----------A program sub-unit, usually used in common by more
(Closed) than one program, which is entered via a transfer from the main program and exits via a transfer back to a selected point in the main program.

SUBROUTINE----------A program sub-unit which is included in the normal
(Open) sequence of a program.

TRACK----------The path around a storage drum, traced out by each head of the drum assembly. Also, the information stored on a given track.

TRANSFER----------To move information from one storage area to another. To depart from the linear sequence of the program instructions by shifting control to another area of the program. This shift is often conditional upon the results of a program test of an indicator word (Conditional Transfer).

UTILITY PROGRAM------A program designed to assist in the full utilization of the computer. Included in this class are Assembly Programs, Compilers, Input/Output Programs, etc.

WORD----------A set or string of symbols which occupies one complete storage register in the computer. This word may be treated as an instruction or as a data word, depending upon the manner of its occurrence in a program.

2 ARITHMETIC FOR PROGRAMMERS

The concept of number is so basic to our everyday life that the average person rarely has occasion to reflect on the many meanings of number, or on the many forms in which numbers are expressed. However, the advent of the electronic computer has necessitated a reconsideration of number systems and their applicability to the conveyance and manipulation of information through electronic circuitry.

Numbers are used, primarily, to denote quantity or amount. We use numbers to state how many pills are in a bottle, or how many pounds of coffee in a bag, or how many miles between here and Tin Cup, Colorado. But there are other ways of using numbers. A number can also be used to represent one member of an ordered set of symbols. A house number, for example, serves to identify a particular house among a number of houses in its set. It further serves to locate that house with respect to these other houses. When a number is used for this dual purpose of location and identification, it is generally referred to as an address. Finally, a number which serves only to identify, but which has no quantitative or locational significance, is usually called a code number.

All three of these aspects of number---quantity, address and identity---are vital to the design and operation of a high speed digital computer. However, the present discussion will concern itself, primarily, with the quantitative aspect of number.

NUMBER SYSTEMS

A Number System may be classified by the number of counting symbols it employs. This number is referred to as the base of the system. The Decimal System is a base-ten Number System; that is, it uses the ten numeric characters, 0 thru 9. It is further characterized by its use of a positional notation. When counting, if one digit position progresses beyond 9, it adds a count in the position immediately to its left. Thus, 9 plus 1 becomes 10; 19 plus 1 becomes 20; 99 plus 1 becomes 100; etc. Each position in a decimal number represents an integral power of 10, so that in the number 456, the number 4 represents 4 times 10^2; the 5 represents 5 times 10^1; and the 6 represents 6 times 10^0. Similarly, the decimal fraction .789 represents 7 times 10^{-1} plus 8 times 10^{-2} plus 9 times 10^{-3}.

Obviously, we are not restricted to a base-ten system of numbers. The same notational structure applies equally well to other bases. And, in fact, it has been determined that today's electronic digital computer functions most efficiently using a binary system for its internal functions. This base-two system follows naturally from the fact that the computer circuitry is made up, in large part, of bi-stable devices.

THE BINARY SYSTEM

The Binary System is, of course, a system which utilizes only two numeric characters, 0 and 1. The principles of counting and arithmetic are exactly the same as for decimal numbers. Counting proceeds in the order, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, etc. Whereas, in the Decimal System every tenth count in a digit position causes a change in the next position to its left; in the Binary System every second count steps the next higher digit position. Each successive position to the left represents the next higher power of two. Thus, the binary number 10110 is equal to 1 times 2^4 plus 0 times 2^3 plus 1 times 2^2 plus 1 times 2^1 plus 0 times 2^0. This is equivalent to the decimal number 22.

Similarly, in a fractional binary number, the positions to the right of the binary point represent successive negative powers of two. Thus, the binary fraction, .10110, represents 1 times 2^{-1} plus 0 times 2^{-2} plus 1 times 2^{-3} plus 1 times 2^{-4} plus 0 times 2^{-5}. This is equivalent to the decimal, 1/2 plus 1/8 plus 1/16, or 11/16.

If we were to perform all our manual computations in binary arithmetic, we would find that, although the individual operations are childishly simple, we would soon be drowning in a sea of 0's and 1's. The decimal number 999,999, if represented in binary, requires 20 digits. The computer, on the other hand, finds it much easier to handle a large number of digit positions than to deal with more than two digit values. In order to facilitate communication between the programmer and the computer, it is desirable to use a number system which provides for brief numerical notation; and one which permits easy conversion to and from the binary system.

THE HEXADECIMAL SYSTEM

If we look carefully at the Binary System, we note that four digit positions are sufficient to hold 16 numerical values from 0 to 15. The RPC-4000 has a word length of 32 bits (binary digits), so that one word can be divided evenly into eight 4-bit groups. If we select, for communication with the computer, a number system with a base of 16, we can represent a full computer word with just eight characters, and each character will correspond exactly with a unique pattern of binary digits.

This base-sixteen number system is known as a Hexadecimal System. It is the system used for all direct communication with the computer. The table of Hexadecimal characters and their decimal and binary equivalents is as follows:

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

BINARY CODED DECIMAL

Frequently encountered in discussions of computers and programming is another, and somewhat different, system of numbers known as Binary Coded Decimal. This is actually a system built on two bases, 2 and 10. Base 2 is used in the binary representation of each of the ten numeric characters, 0 thru 9. Normally, the binary numbers, 0000 thru 1001 are used, and each 4-bit group signifies one decimal character. The positional notation used is the same as for decimal numbers in that each successive 4-bit group proceeding to the left of the units position represents the next higher power of ten. Thus the decimal number in binary is written 101110001.

The usefulness of this number system is in providing a convenient form in which to process numerical data through the Input/Output equipment. Its use, however, requires Input and Output program routines which will make the conversion to and from a true binary representation.

CONVERSION BETWEEN SYSTEMS

In working with computers and programs, it frequently becomes necessary to perform a manual conversion from one number system to another. This is particularly true in debugging, where a word displayed on the scope face must often be verified against a symbolically written program word.

Conversion of a decimal integer into its binary and hexadecimal equivalent may be accomplished by a system of successive divisions of the decimal integer by the base of the system into which we are converting. For example, consider the decimal number, 229. If we divide by the hexadecimal base, 16, we get 14, with a remainder of 5. The 5 represents the least significant digit of the hexadecimal number we are computing. If we now divide the integral quotient, 14, by 16, we get 0, with a remainder of 14. The 14, expressed hexadecimally by the character, E, represents the most significant digit of the desired number, so that the hexadecimal equivalent of 229 is E5. To avoid confusion when working with number conversions, the base of a number is usually represented by a subscript, so that the above numbers would be more clearly expressed as 229_{10} and $E5_{16}$. Let us consider another example, this time showing the conversion in tabular form. The decimal integer 1386 would be converted as follows:

DECIMAL NUMBER	÷	16	=	QUOTIENT	with	REMAINDER
1386_{10}				86_{10}		(least significant) A_{16}
86_{10}				5_{10}		6_{16}
5_{10}				0		(most significant) 5_{16}

The hexadecimal equivalent of 1386_{10} is $56A_{16}$. Note that the "A" in the above example is used as a hexadecimal numeral, and is the equivalent of a decimal 10.

Decimal integers may be converted to binary in exactly the same fashion. Conversion of 52_{10} to binary proceeds as follows:

DECIMAL NUMBER	÷	2	=	QUOTIENT	with	REMAINDER
52_{10}				26_{10}		(least significant) 0_2
26_{10}				13_{10}		0_2
13_{10}				6_{10}		1_2
6_{10}				3_{10}		0_2
3_{10}				1_{10}		1_2
1_{10}				0_{10}		(most significant) 1_2

The binary equivalent of 52_{10} is shown to be 110100_2. With a little practice, a binary number can be readily converted to decimal by inspection or by referring to a table of powers of two. Converting the above binary value to decimal is just a matter of adding 2^5 (32), 2^4 (16), and 2^2 (4) to arrive at the value, 52_{10}. Fractional values may be similarly handled by using negative powers of two.

Conversion of a decimal fraction into hexadecimal or binary may be accomplished by a system of successive multiplications by the base into which we are converting. For example, consider the following conversion of $.912_{10}$ into hexadecimal.

DECIMAL NUMBER	X 16 =	PRODUCT	with	INTEGRAL CARRY	= HEXADECIMAL (most significant)
$.912_{10}$		14.592_{10}		14_{10}	E_{16}
$.592_{10}$		9.472_{10}		9_{10}	9_{16}
$.472_{10}$		7.552_{10}		7_{10}	7_{16}
ETC.					

In this process, the first integral carry represents the most significant digit of the hexadecimal fraction, and the fraction, $.912_{10}$ converts to $.E97_{16}$, computed to 3 places.

SCALING

The numbers $.752_{10}$, 7.52_{10}, 75.2_{10}, and 752_{10} are identical in appearance except for the placement of an insignificant looking symbol known as a decimal point. Yet the use of this symbol can make a huge difference in the meaning of the number in which it is placed. When we work with numbers in this fashion, we arbitrarily place this point in such a way as to establish a desired magnitude for the number we wish to represent. The digits to the left of this point represent integral values, and the digits to the right represent fractional values. In order to perform valid arithmetic operations with these numbers, we must necessarily be cognizant of the location of the point in all our operands.

The RPC-4000 computer word is 32 binary digits in length. For a data word, these bits provide for a sign bit in position 0, and 31 magnitude bits. The principles of scaling, outlined above with respect to decimal arithmetic, apply also to the binary values used by the computer. The binary point does not actually exist within the computer. It exists only

in the mind or on the paper of the programmer. It is his responsibility to control the placement of values in the computer word so that these implied binary points will fall in the proper positions to produce valid arithmetic results.

The binary point location between bits 0 and 1 of the computer word, referred to as the machine point, serves as a reference point in specifying the scale factor for a value in the word. The symbol, "q", has been established by convention to denote the placement of the implied binary point with respect to the machine point. If a value is entered into the computer at a "q" of 5, the bits in positions 1 thru 5 represent the integral portion of the value, and the bits in positions 6 thru 31 represent the fractional portion.

To perform a valid addition or subtraction in the computer, the binary points must be lined up the same as we would line up the points in performing these operations with pencil and paper. That is, the operands must exist in the computer at the same "q". A number at a "q" of 12, added to another number at a "q" of 12, will produce a sum which is also at a "q" of 12.

In multiplication, however, the points need not be lined up, but must be located so as to produce a product at the required "q". The "q" of the product of a multiplication is the sum of the "q's" of the multiplicand and the multiplier. A number at a "q" of 6, multiplied by a number at a "q" of 3, will produce a product at a "q" of 9. An exception to the above is the command MPT (Multiply by Ten), in which the "q" of the product is the same as the "q" of the multiplicand. (See Page 44).

The rules for performing a division in the computer are equally simple, but require one additional precaution on the part of the programmer. The "q" of the quotient is determined by subtracting the "q" of the divisor from the "q" of the dividend. The value 6, at a "q" of 3, divided by the value 2, at a "q" of 2, should produce a quotient, 3, at a "q" of 1. But the value, 3, cannot be held at a "q" of 1, so that a condition known as a "divide check" occurs. This produces an overflow out of the accumulator which will turn on an indicator, known as the Branch Control. To avoid this situation, the programmer must scale his operands so that the quotient will fit in the accumulator. In the above example, shifting the dividend 6, to a "q" of 4, and then dividing by 2 at a "q" of 2, will produce a quotient of 3, at a "q" of 2, which can be contained without overflow. Considered from the standpoint of the computer word itself, and disregarding the implied binary point location, the divisor must always exceed the dividend to perform a valid divide operation.

The means for controlling the scaling of values in the computer are provided by a shift instruction, which permits the programmer to adjust the "q" of any value by shifting the accumulator contents right or left by a prescribed number of bit positions. (See Page 41). It is important, in coding a program, to note on the coding sheet the "q" values of all operands.

3 THE CENTRAL COMPUTER

The basic unit of information in the computer is referred to as a word. The computer word in the RPC-4000 consists of 32 binary digits, commonly called "bits". This 32 bit pattern of information has a meaning dependent upon the context in which it is used. That is to say, that the same word from memory may be used as an instruction or as an arithmetic operand, and this usage is determined by the manner in which the program is written.

When used as numerical data, a word is considered as consisting of a sign bit in the left-most position, followed by 31 magnitude bits. Preceding a DIV (Divide) instruction or following a MPY (Multiply) instruction the data word in the LOWER Accumulator is considered as an extension of the UPPER and contains magnitude bits in bit positions 0 thru 30, bit 31 being disregarded.

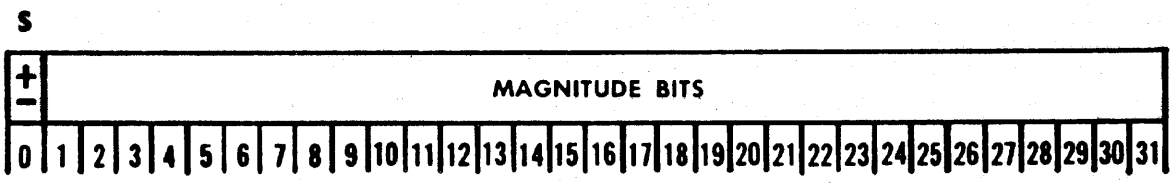


FIGURE 2. RPC-4000 Data Word Format

When used as an instruction, a word is considered as consisting of a command in bits 0 thru 4, an operand or an operand address in bits 5 thru 17, the next instruction address in bits 18 thru 30, and the index tag in bit 31.

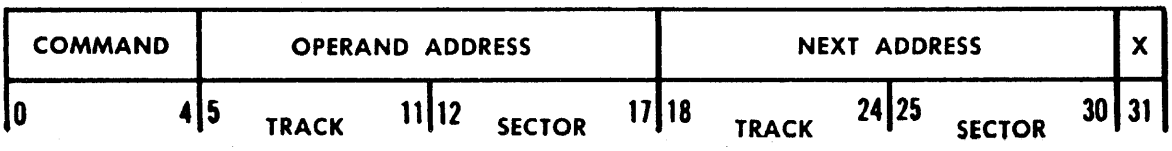


FIGURE 3. RPC-4000 Instruction Word Format

In certain cases it is convenient to consider the 32 bits of a word as eight hexadecimal characters. This is particularly true in entering information manually into the computer or in analyzing certain computer outputs.

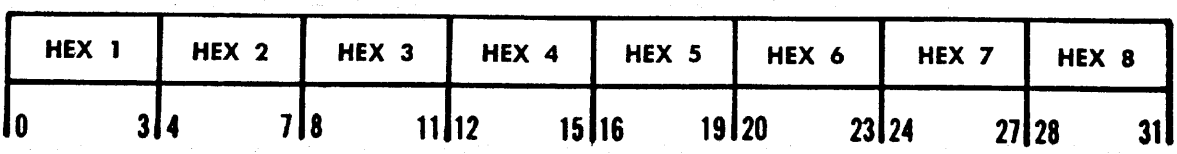


FIGURE 4. Hexadecimal Word Format

THE MEMORY DRUM

Information is stored in the RPC-4000 on a magnetic drum which rotates at the rate of 3600 revolutions per minute. This information will consist of both data words and instruction words. Each individual bit of each word stored in the computer is in the form of a discrete magnetized spot on the iron oxide coated drum surface. Each bit can exist in one of two magnetic states representing the binary values, 0 (zero) and 1 (one). All information in the computer, including all memory storage and all working registers, is represented in this manner.

The computer words are arranged in parallel bands around the drum, each band containing 64 word positions. Associated with each of these bands is a magnetic read/write head (those bands used for double access each have two read/write heads.) The band of information traced out by any given head is referred to as a track. Each of the 64 word positions around the circumference of the drum is referred to as a sector.

The RPC-4000 memory consists of 128 tracks (numbered 000 thru 127) and 64 sectors (numbered 00 thru 63). Hence, the location of any word in memory can be specified by its track and sector number. This number is known as the address of the word in question.

Tracks 000 thru 122 of the magnetic drum are the Main Memory storage area for the RPC-4000. Each track has 64 associated sectors, each of which contains one computer word. The total storage capacity of Main Memory is 7872 words. Any word in Main Memory may be referenced by specifying its track and sector position. Thus, 09843 refers to the word whose address is Track 098, Sector 43.

Maximum access time for a word in Main Memory is the time required for one complete drum revolution, or approximately 17 milliseconds. Average access time is approximately 8.5 milliseconds.

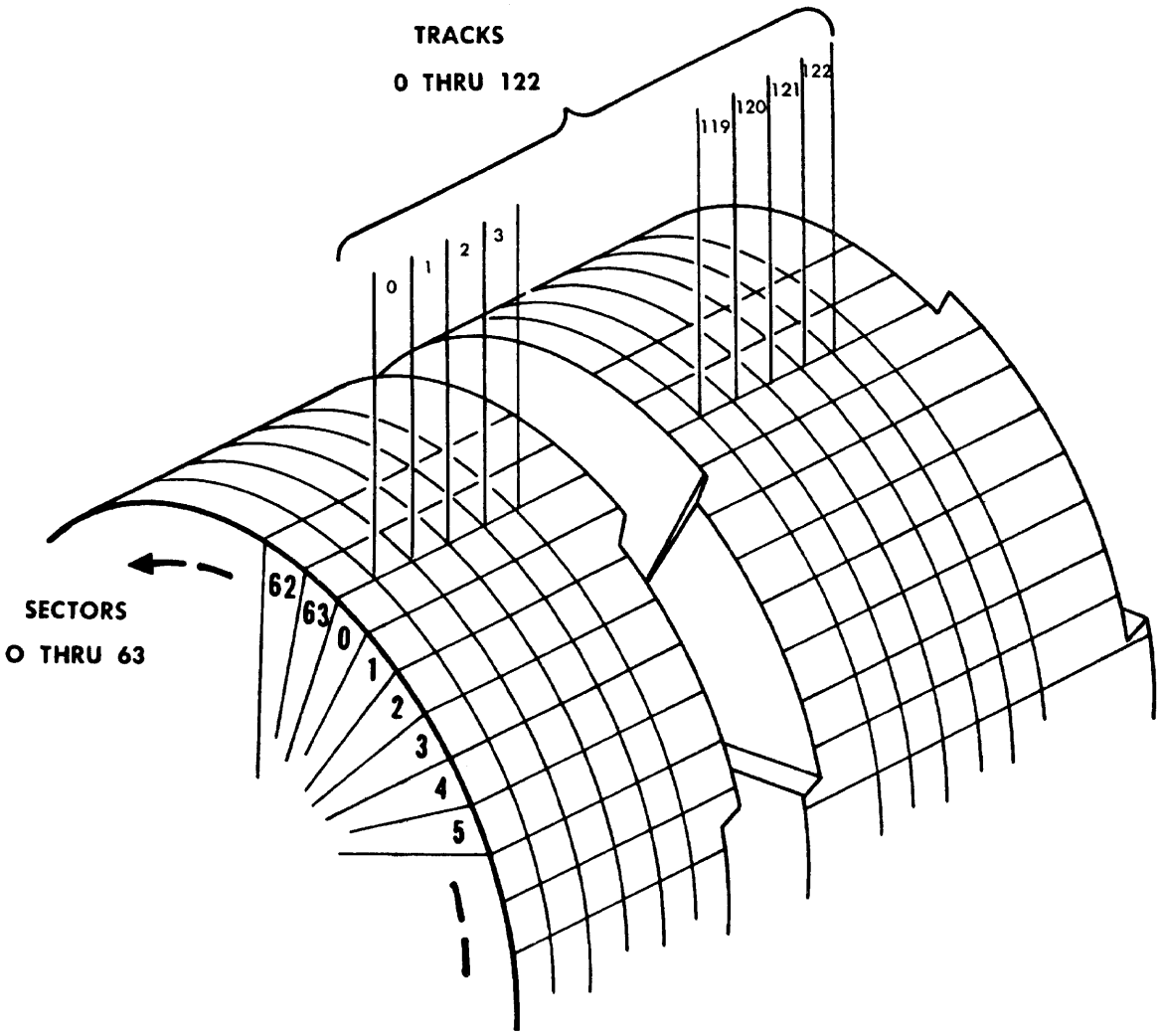


FIGURE 5. RPC-4000 Main Memory Storage

DOUBLE ACCESS TRACKS

In addition to the Main Memory just discussed, there are two bands of memory storage on the drum for which double access is provided. Each of these bands has two read/write heads, addressed by their own individual track numbers. The heads addressed as Track 123 and Track 125 have a common storage band. The Track 125 head is displaced so as to be 16 word times later than the Track 123 head. Therefore, both heads refer to the same set of words, and the word which is addressed by 12301 is the same word as that addressed by 12517.

Likewise, the heads addressed as Track 124 and Track 126 have a common set of word positions. In this case, the Track 126 head is displaced so as to be 24 word times later than the Track 124 head. The word addressed by 12401 is the same word as that addressed by 12625.

Random access time for a word from the Double Access storage area is approximately 17 milliseconds, the same as for a word from Main Memory. However, a second access to the same word requires only about 4 milliseconds for Track 123/125 or 6 milliseconds for Track 124/126. Total Double Access storage capacity is 128 words.

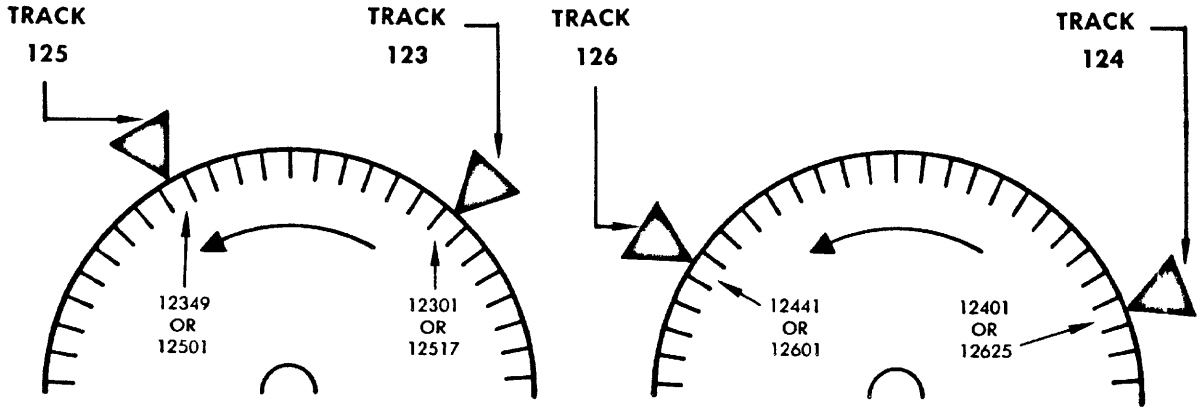


FIGURE 6. Double Access Storage Tracks

THE RECIRCULATING TRACK

There are 8 additional words of memory storage on the drum in what is known as the Recirculating Track. The associated read/write head is referred to as Track 127, and this track provides a special rapid access storage area. The rapid access capability derives from the fact that these 8 words are repeated 8 times around the drum. This is accomplished in the following manner.

In addition to, and flanking the normal read/write head are a separate read head and write head. The read head is placed so as to trail the write head by 8 words. That is to say, that a word position on the drum will reach the write head 8 words before reaching the read head. Consequently, as each word is read by the read head, it is immediately rewritten 8 word positions farther back on the drum. Likewise, any new word value written on the drum by the normal read/write head will be picked up when it reaches the read head and recirculated.

Since every eighth word around the drum on Track 127 is identical, sector addresses will be modulo 8. The addresses 12701, 12709, and 12717 will all refer to the same word inasmuch as the sector addresses differ by multiples of 8. With 8 accesses provided for each word every drum revolution, maximum access time for the Recirculating Track is approximately 2 milliseconds, with an average access time of about 1 millisecond.

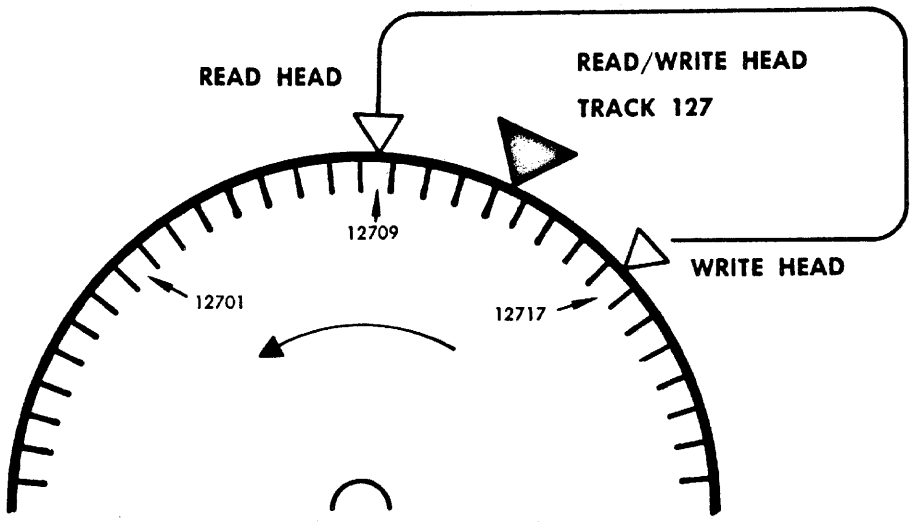


FIGURE 7. The Recirculating Track

THE SECTOR REFERENCE TIMING TRACK

Although this track cannot be read directly or modified by the programmer, some explanation of its function may be helpful. The Sector Reference Timing Track contains the Sector Reference numbers 00 thru 63, which are permanently pre-recorded at the time of manufacture. The track has a read head only, which serves to locate, by reference to this Timing Track, any sector address on the drum which is called for in a program.

The Sector Reference numbers on the Timing Track are accessible to a program only in the case of a compare instruction (CME or CMG) which is executed in Repeat Mode (See Pages 50 & 51). During a repeated compare instruction the contents of this track are continually copied into bits 25 thru 30 of the INDEX Register until a successful comparison is found, or until completion of the instruction, at which points any further copying is inhibited. The Sector Reference number is always one greater than the sector being compared. Consequently, following a successful comparison (Branch Control On), the sector address of the memory word which compares can be determined by subtracting 1 from the Sector Reference value in the INDEX Register.

THE WORKING REGISTERS

A register may be described as the hardware for storing a single computer word. Those registers which are used to perform the various arithmetic and control functions within the computer are the Working Registers. There are four such registers in the RPC-4000, each in the form of a recirculating track on the drum with a one-word spacing.

These four Working Registers and associated logic together comprise what is referred to as the Computing Control Unit. As such, they perform all internal data processing and control inputs and outputs to and from the computer. The registers are known individually as:

U----------UPPER Accumulator
L----------LOWER Accumulator
C----------COMMAND Register
X----------INDEX Register

THE UPPER ACCUMULATOR

Generally speaking, the UPPER Accumulator is that register in the Computing Control Unit which holds the computer word to be operated upon. At the completion of an arithmetic or logical operation, it will normally hold the result of the operation.

The UPPER Accumulator is the primary working resiter. It can receive information from, or send information to any register in memory, and any working register except the COMMAND Register. For most instructions these are full word transfers. However, information may be shifted from the UPPER to the LOWER Accumulator, or from the LOWER to the UPPER Accumulator on a bit-by-bit basis.

For any arithmetic operation, the contents of the UPPER Accumulator are considered as consisting of a sign bit followed by 31 magnitude bits and the operation will be performed according to the normal law of signs. Preceding a Multiply (MPY) instruction, the UPPER will contain the multiplicand. Following a Multiply instruction, the UPPER Accumulator will contain the most significant half of the double word product. Preceding a Divide (DIV) instruction, the UPPER will contain the most significant half of the double word dividend. Following a Divide instruction, the UPPER will contain the quotient.

THE LOWER ACCUMULATOR

The LOWER Accumulator may be thought of as a supplement to the UPPER, providing an alternate data handling register, and serving as an extension to the UPPER for bit manipulation and extra precision arithmetic. Its contents may be added to, subtracted from, and transferred direct to memory, in the same way that these operations are performed with the UPPER.

Before a Divide (DIV) instruction, the LOWER Accumulator will contain the least significant half of the dividend. Following a Divide (DIV) or a Divide Upper (DVU) instruction, the LOWER Accumulator contains the

remainder. Following a Multiply (MPY) instruction, the LOWER Accumulator contains the least significant half of the product.

When combined with the UPPER Accumulator to form a double length word, bit "0" of the LOWER Accumulator is considered as a data bit. Otherwise, it is considered as a sign bit.

Through the use of a modifier bit in the Exchange (EXC) instruction, the LOWER Accumulator can be set to eight words instead of one. These eight words function as individual accumulators, but are subject to certain restrictions in their use. For further discussion of this mode, see Page 24.

THE COMMAND REGISTER

The COMMAND Register is that register which holds an instruction word during the time that it is being interpreted and operated by the computer. The instruction word consists of four logical parts or fields, each of which is considered when an instruction is performed:

1. The Command Field (bits 0 thru 4)---This field in the COMMAND Register will contain the numerical code representing the operation to be performed. It serves only to identify the required operation and has no quantitative significance.

2. The Data-address Field (bits 5 thru 17)---For most instructions, this field will contain a memory address signifying the location of the operand. For the transfer instructions TBC and TMI, it will contain the address of the next instruction in the event of an active transfer. For some instructions, this field will contain a set of logical modifiers which serve to further define the operation of a basic command. And for a few instructions, the <u>contents</u> of this field are used as the operand.

3. The Next-address Field (bits 18 thru 30)---This field will contain a memory address signifying the location of the instruction which is to operate immediately following the one currently in the COMMAND Register.

   The only exception to this is that, upon encountering an active transfer instruction, the next instruction will be taken from the address specified in the Data-address Field.

4. The Index Tag Field (bit 31)---A "1" bit in this field will cause the computer to apply indexing to the instruction when it is placed in the COMMAND Register. This means that the Data-address Field is incremented by the contents of bits 5 thru 17 of the INDEX Register. If the Index Tag is set to "0", indexing is not applied, and the Data-address field in the COMMAND Register will be identical with the corresponding instruction in memory.

THE INDEX REGISTER

The INDEX Register performs several important functions in the RPC-4000. Its primary use is for address modification and, for this purpose, bits 5 thru 17 of the INDEX Register serve to hold a value by which the Data-address Field of an instruction may be incremented. This incremental value may be placed in the INDEX Register by means of a Load Index (LDX) instruction, and can be used by including an Index Tag in the appropriate instruction.

Bits 18 thru 24 of the INDEX Register are used to hold the Repeat Count for any instruction which is operated in the Repeat Mode. (See Page 25). This count is loaded by a Load Count (LDC) instruction immediately preceding the instruction to be repeated.

Bits 25 thru 30 of the INDEX Register are used in conjunction with the Compare Memory Equal (CME) and the Compare Memory Greater (CMG) instructions when these instructions are executed in the Repeat Mode. At the beginning of the instruction, the Sector Reference Timing Track is copied into bits 25 thru 30 of the INDEX Register and this copying occurs during each iteration until a successful comparison is made or until the specified number of repeats is completed. If and when a successful comparison occurs, any further copying of the Timing Track is inhibited. The INDEX Register will then contain a value one sector greater than the sector location of the memory word which compared successfully.

In addition to the above functions, the full word INDEX Register may be used as a rapid access storage location and may be exchanged with the UPPER Accumulator through the use of the Exchange (EXC) instruction.

THE BRANCH CONTROL

Although it is not a register, there is another information handling device within the computer which should be mentioned here. This is an internal flip-flop consisting of only one bit and known as the Branch Control. It is automatically turned on when an overflow condition oc-

curs. It may be set by the instructions, SNS, CXE, CME, and CMG. It may be sensed by the instruction, TBC (Transfer on Branch Control).

PROGRAMMED OPERATING MODES

There are certain modifications which can be made under program control, which will alter the normal operating procedures for the Central Computer. They are concerned, for the most part, with the processing of multi-word blocks of information or tables of associated data. The first of these modifications provides the ability to change the form of the LOWER Accumulator from one word to eight words in eight individual LOWER Accumulators. The second enables automatic repetition of an instruction for a selected number of times using consecutive storage locations.

THE EIGHT-WORD LOWER ACCUMULATOR

The LOWER Accumulator can be set to eight words or back to one word by means of two control bits which are a part of the Exchange (EXC) instruction word. The effect of setting the LOWER to eight words is that, instead of the same word being recirculated into all 64 word positions, there are eight separate words considered, each of which is recirculated into eight word positions. In this respect, the eight-word LOWER is the same format as the recirculating memory, Track 127.

When an instruction involving the eight-word LOWER is executed, the accumulator word used will correspond to the Data-sector of the instruction word. To put it another way, if we let L_0, L_1, L_2, ... L_7, designate the eight LOWER Accumulators, then an instruction with a Data-address of 10300 would use or affect L_0, as would 10308, 10316, etc. Likewise, an instruction with a Data-address of 02703, would use or affect L_3, as would 05403, 06711, 11819, etc.

If overflow occurs in a LOWER Accumulator word, it has no effect on the other seven words of the LOWER, nor does it affect the UPPER Accumulator. It does, however, turn on the Branch Control.

The eight-word LOWER Accumulator can be used with the Input (INP) instruction to receive up to 64 4-bit characters or 42 6-bit characters. Inputs enter into the least significant end of L_0, and each succeeding character causes the preceding characters to be shifted left until all eight accumulators are filled. If the number of input characters exceeds the capacity of the eight-word LOWER, characters will be lost out of the most significant end of L_7.

Those instructions concerned with the LOWER which require more than one word time for execution (MPY, DVU, DIV, SRL, SLC,) should <u>not</u> be executed while the LOWER is in eight-word format.

THE AUTOMATIC REPEAT MODE

The execution phase of an instruction operated in this mode may be extended to any desired number of word times, not to exceed 128. A word time is that time required for one sector of the drum to pass beneath the read/write head. It is approximately .25 milliseconds. Thus, the number of repetitions of an instruction will be a function of the Repeat Count and the word times required to complete one execution. Those instructions which are appropriate for repeated execution require only one word time for command execution and, therefore, the number of repetitions beyond the initial execution will be equal to the specified Repeat Count.

The Repeat Mode is initiated by a Load Count (LDC) instruction, which places the Repeat Count in the Repeat Control field of the INDEX Register (bits 18 thru 24). The instruction immediately following the LDC instruction will be the only one affected. (See exceptions below). It will be placed in the COMMAND Register, and where appropriate, the drum will be searched for its Data-address as in normal operation. But its execution phase will be continually repeated until the Repeat Count in the INDEX Register runs out. For each repetition, the Data-address used will be the one beneath the head at the time of execution. When this operation continues beyond Sector 63, it proceeds with Sector 00, 01, etc., of the same track. Track switching is not possible during a Repeat Mode operation.

The Repeat Count in the INDEX Register does not define the number of repetitions of an instruction whose execution phase requires more than one word time. These instructions, when executed in the Repeat Mode, produce results which are not true repetitions of the execution phase, and this usage is to be avoided.

The primary uses of the Repeat Mode are with the arithmetic instructions for block processing or block transfers, and with the compare instructions, CME and CMG, for table search.

A special case occurs when a conditional transfer instruction, TBC, or TMI, is operated in the Repeat Mode. If the transfer is inactive, no action is taken and the execution phase merely delays proceeding to the next instruction until the Repeat Count runs out. If the transfer is active, the normal execution phase is never reached. Consequently, the Repeat Count remains intact and causes the repetition of the instruction to which the transfer is made. This is the only instance in which the repeated instruction does not immediately follow the LDC instruction.

INSTRUCTION SEQUENCING AND TIMING

The sequence in which instructions are executed is controlled by addresses contained within the instruction word. A complete instruction

cycle extends from the beginning of the memory search for the instruction word to the beginning of the search for the next instruction word.

This cycle consists of four phases as follows:

Phase 1 - Search for memory location specified in Next-address Field of COMMAND Register --- 1 to 64 word times.

Phase 2 - Transfer contents of this location to the COMMAND Register --- 1 word time.

Phase 3 - Search for memory location specified in Data-address Field of COMMAND Register --- 1 to 64 word times.

Phase 4 - Execution of instruction --- Basically, 1 word time but certain instructions will include a Sub-phase 4 (a) which extends this execution time.

Except for EXC (Exchange) and MPT (Multiply by Ten), those instructions which do not require a memory search for data will require only 1 word time in Phase 3. Those instructions are:

HLT	(Halt)	SLC	(Shift Left and Count)
SNS	(Sense)	PRD	(Print from Data-address)
CXE	(Compare Index Equal)	PRU	(Print from UPPER)
LDX	(Load Index)	INP	(Input)
SRL	(Shift Right or Left)		

Phase 3 for EXC and MPT will depend on the Data-sector value of the instruction word, so as to allow an association with a particular word of the Eight-word LOWER Accumulator.

The following instructions will include a Sub-phase 4 (a) in addition to the 1-word Phase 4, which will extend their execution times as indicated.

DVU (Divide Upper) - 67 word times
DIV (Divide) - 67 word times
SRL (Shift Right or Left) - 4 word times plus 1 word time for each bit position shifted.
SLC (Shift Left and Count) - 4 word times plus 1 word time for each bit position shifted.
MPY (Multiply) - 67 word times.

All other instructions require one word time for execution except for INP (Input), PRD (Print from Data-address) and PRU (Print from UPPER), which are dependent on the speed of the selected Input/Output devices.

4

THE INSTRUCTION LIST

The basic instructions which make up the RPC-4000 Instruction List are described in the following pages. These are "machine language" instructions, each of which specifies a particular action to be taken by the computer. Pseudo-instructions for communication with the ROAR assembly program are not included in this section.

A drawing of the Instruction Word format accompanies each description. This drawing contains the numerical Command code and illustrates the field allocation for each component of the Instruction. The mnemonic designator for each Command is shown in bold type at the top of each description.

In many instances, diagrams have been included to support and clarify the textual explanations of the actions taken by the Instructions. These diagrams indicate the transfer and modification of the contents of the various registers which are involved in the instruction operation.

The "minimum time" referred to in these pages includes the minimum time to complete all four phases of the operation, from the beginning of the search for the Instruction Word to the beginning of the search for the next Instruction Word.

HLT HALT

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
00	000	(ANY)	(NEXT ADDRESS)		0/1
0 0000 4	5 11	12 17	18 24	25 30	31

The computer will halt with the Instruction Word in the COMMAND Register.

If indexing is specified, it should be noted that any index value which results in a non-zero value in the D-track field will effectively produce a SNS instruction. (See next page)

Minimum time---------4 word times
Overflow------------Not a factor
Branch Control------Not affected
Registers affected--None

SENSE **SNS**

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
00	1 ⟶ 127	(ANY)	(NEXT ADDRESS)		0/1
0 0000 4	5 11	12 17	18 24	25 30	31

The initial action of the instruction is to turn off the Branch Control.

This instruction will sense the condition of certain manual switches as related to bits 6⟶11 of the D-track field. If one or more D-track bits correspond to a depressed Sense Switch, the instruction will turn on the Branch Control.

If bit 5 of the D-track field is set to 1, it causes the Branch Control to be turned on if no input device is selected, or if any selected input or output device is not ready.

Note that any index value which modifies the value of the D-track will alter the logical correspondence of the D-track bits and the Sense Switches.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control-------Conditionally set "ON" or "OFF"
Registers affected---None

DATA-TRACK/SENSE SWITCH CORRESPONDENCE

D-track Value	Bit Pattern	Sense Switch
1	0000001	1
2	0000010	2
4	0000100	4
8	0001000	8
16	0010000	16
32	0100000	32

CXE

COMPARE INDEX EQUAL

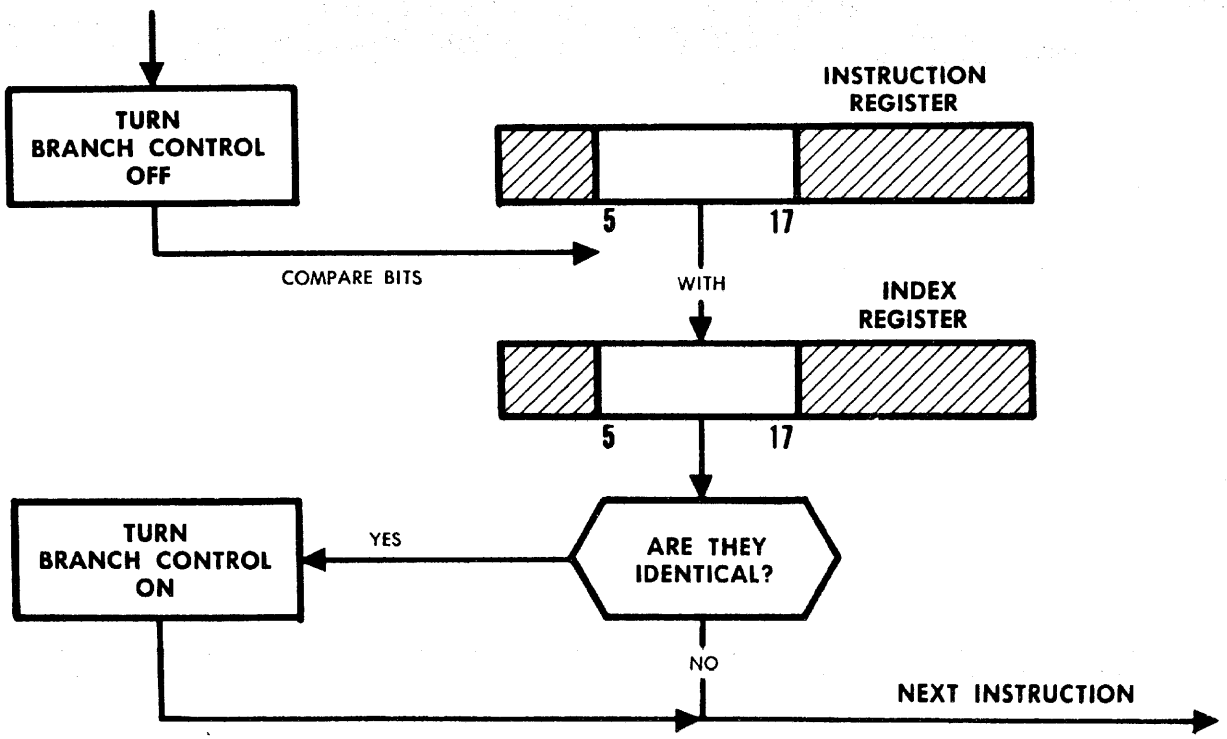
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
01	(DATA VALUE)		(NEXT ADDRESS)		0/1
0 0001 4	5	11 12 17	18	24 25	30 31

The initial action of this instruction is to turn off the Branch Control. The Data Value in the instruction is then compared with bits 5 → 17 of the INDEX Register. If there is a one-to-one correspondence, the Branch Control is turned on. If not, the Branch Control remains off.

If indexing is specified for this instruction, a Data Value of zero will turn on the Branch Control, regardless of the index value. This occurs because the zero Data Value, when indexed, becomes identical with the Index Value. Conversely, any Data Value other than zero will turn off the Branch Control, regardless of the Index Value, inasmuch as any non-zero value, when indexed, becomes greater than the Index Value.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Conditionally set "On" or "Off"
Registers affected---None



RESET AND ADD TO UPPER **RAU**

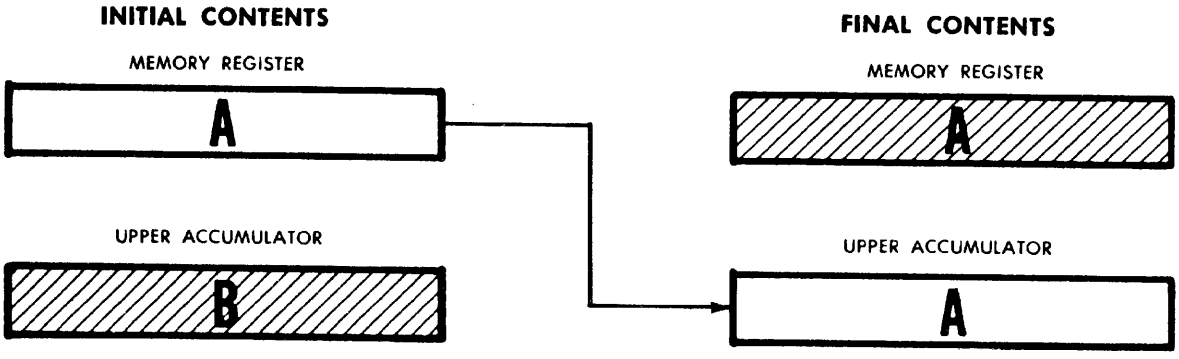
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
02	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 0010 4	5 11	12 17	18 24	25 30	31

The contents of the memory location specified by the Data-address will replace the current contents of the UPPER Accumulator.

The Data-address may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Register affected----UPPER Accumulator



RAL

RESET AND ADD TO LOWER

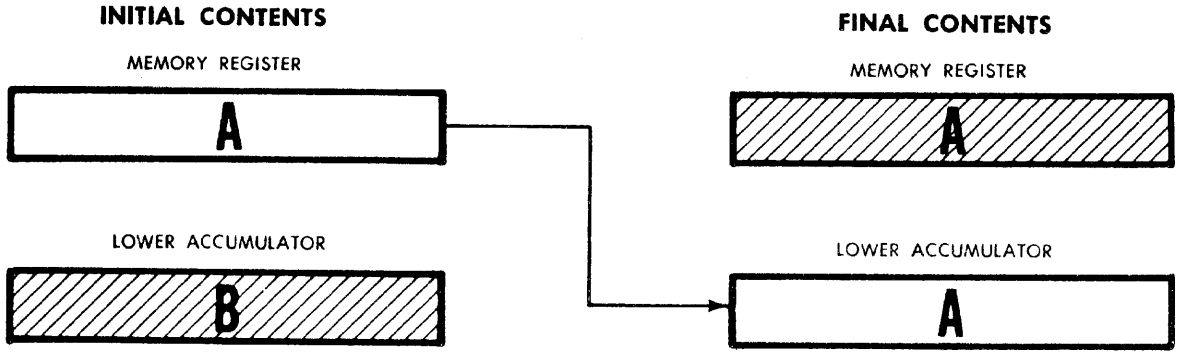
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
03	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 0011 4	5	11 12 17	18	24 25 30	31

The contents of the memory location specified by the Data-address will replace the current contents of the LOWER Accumulator.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control-------Not affected
Registers affected---LOWER Accumulator



STORE ADDRESS FROM UPPER **SAU**

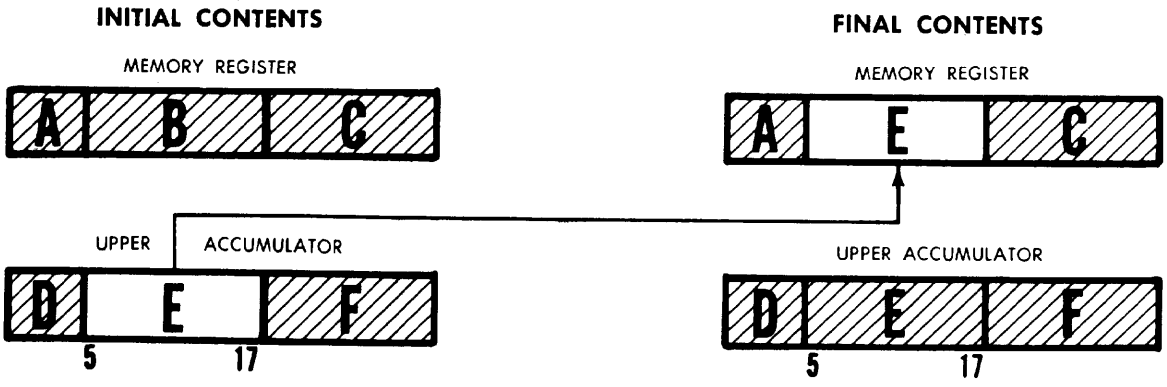
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
04	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 0100 4	5	11 12 17	18	24 25 30	31

Stores bits 5⟶17 of the UPPER Accumulator into bits 5⟶17 of the memory location specified by the Data-address, replacing the current contents of these bits. The remainder of the memory word is left unchanged.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control-------Not affected
Registers affected---Specified memory location



MST MASKED STORE

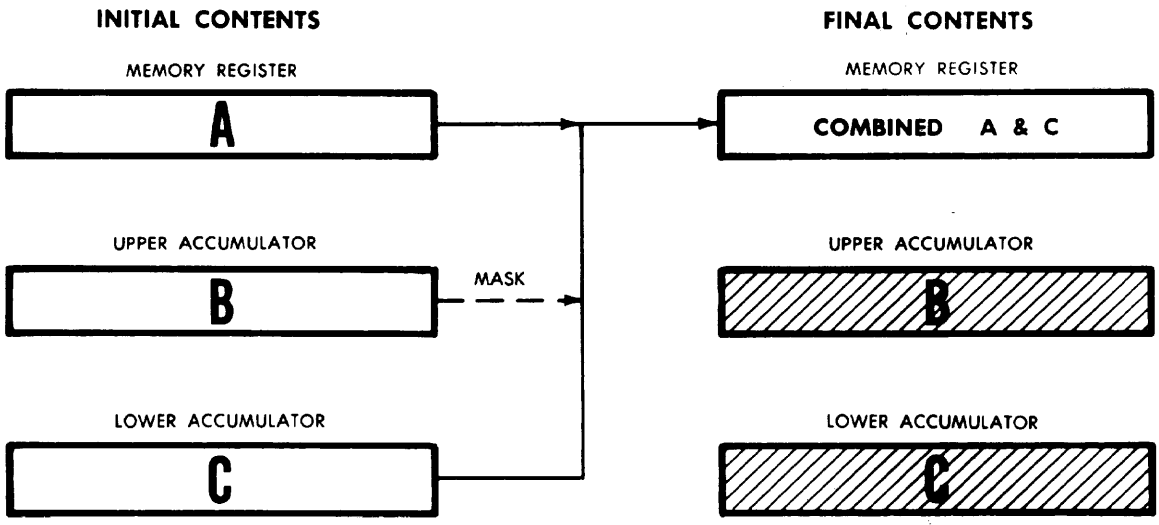
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
05	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 0101 4	5	11 12 17	18	24 25 30	31

Stores selected bits from the LOWER Accumulator into the memory location specified by the Data-address, as masked by the UPPER Accumulator. Where the UPPER Accumulator contains 1's, stores the corresponding LOWER Accumulator bits into the memory word. Where the UPPER Accumulator contains 0's, leaves the corresponding memory word bits unaltered.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control-------Not affected
Registers affected---Specified memory location



LOAD COUNT **LDC**

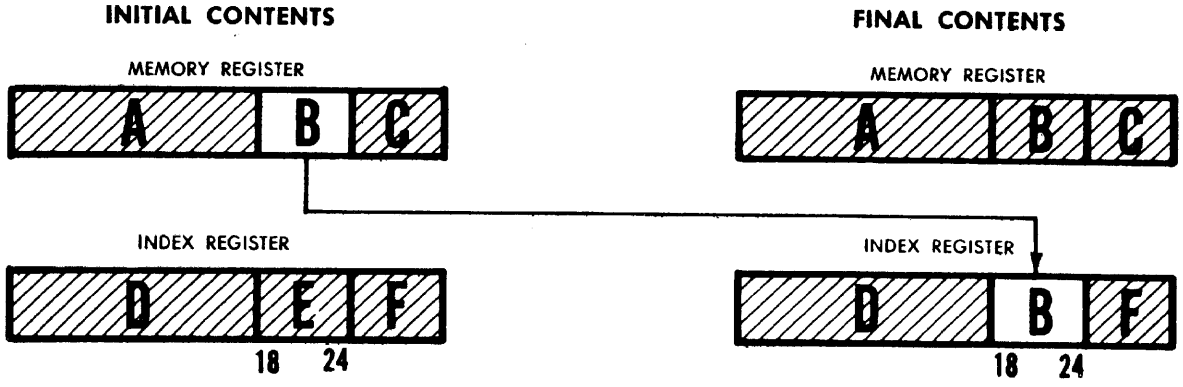
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
06	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 0110 4	5	11 12 17	18	24 25 30	31

Loads bits 18⟶24 of the memory location specified by the Data-address into bits 18⟶24 of the INDEX Register, replacing the current contents of these bits. The remainder of the INDEX Register is left unchanged.

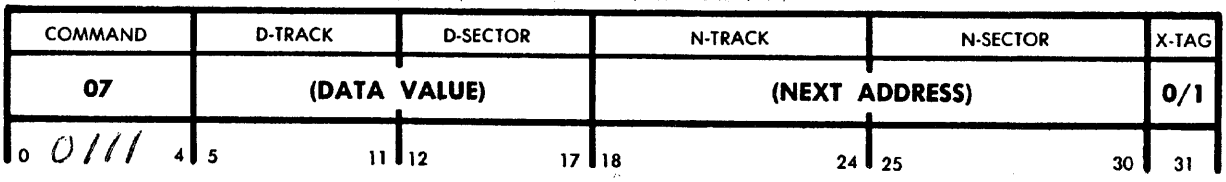
Causes the next instruction to be executed in the Repeat Mode; that is, the instruction contained in the memory location specified by the Next-address will be repeated as many times as the number placed in bits 18⟶24 of the INDEX Register. If the next instruction is an active TBC or TMI, the repeated instruction will be that to which the transfer is made (See page 25).

Minimum time----------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---Index Register



LDX LOAD INDEX

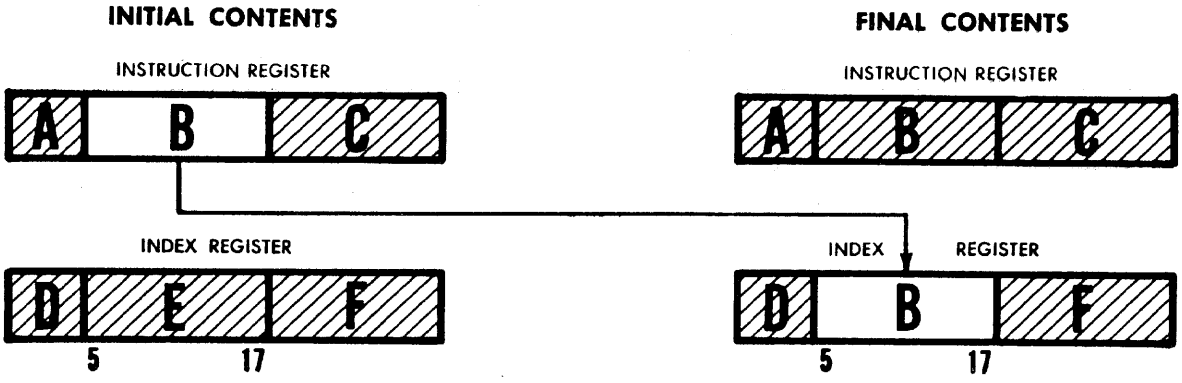
INSTRUCTION WORD FORMAT



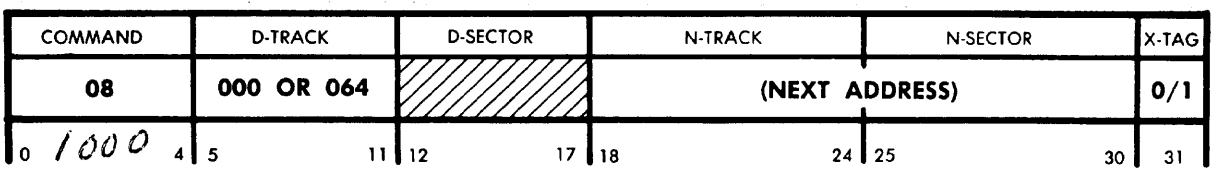
Loads the Data Value from the Instruction Word into bits 5⟶17 of the INDEX Register, replacing the current contents of these bits. The remainder of the INDEX Register is left unchanged.

The Data Value may be modified by indexing. This will serve to increment the index value by the number in the Data Value Field.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control-------Not affected
Registers affected---INDEX Register



INSTRUCTION WORD FORMAT



If the D-track of the Instruction Word contains 000, reads 4-bit characters into the accumulator(s). If the D-track contains 064, reads 6-bit characters into the accumulator(s).

If the LOWER Accumulator is set at 1-word length, the characters will be read into the double length accumulator (combined UPPER and LOWER). If the LOWER Accumulator is set at 8-word length, the characters will be read into the LOWER Accumulator only. Read-in begins with the low order character position of the low order accumulator word and, as subsequent characters are read in, the existing accumulator contents are shifted left one character position.

The D-track value may be modified by indexing. However, care should be taken that this does not result in a value other than 000 or 064.

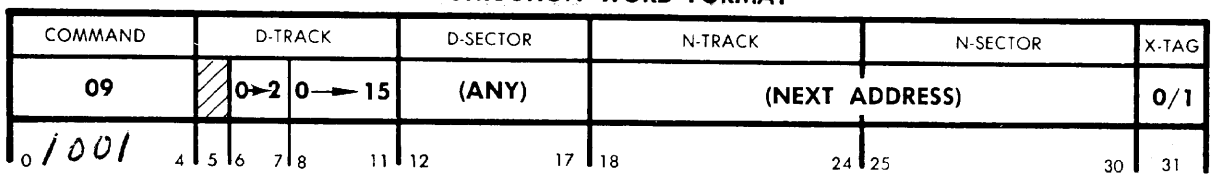
If the D-track field of the INP instruction contains a value other than 000 or 064, the character which enters the LOWER Accumulator will be the logical sum of the incoming tape character and the corresponding bits from the D-track field of the instruction word.

Minimum time---------Dependent on the number of characters read
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---UPPER and/or LOWER Accumulators

EXC

EXCHANGE

INSTRUCTION WORD FORMAT



This instruction performs one or more of the following functions, in accordance with the logical bit pattern in the D-track field.

If bit 6 = 1,----------Set the LOWER Accumulator to 1-word length.
If bit 7 = 1,----------Set the LOWER Accumulator to 8-word length.
(If both = 1, the current state of the LOWER will be reversed).
If bit 8 = 1,----------Replace the contents of the UPPER Accumulator with the contents of the INDEX Register.
If bit 9 = 1,----------Replace the contents of the INDEX Register with the contents of the UPPER Accumulator.
If bit 10 = 1,----------Replace the contents of the UPPER Accumulator with the contents of the LOWER Accumulator.
If bit 11 = 1,----------Replace the contents of the LOWER Accumulator with the contents of the UPPER Accumulator.

(Any or all of the above exchanges may be executed with the same instruction. If bits 8 and 10 are both set, the UPPER Accumulator will receive the logical sum of the contents of the INDEX Register and the LOWER Accumulator).

The D-track and D-sector values may be modified by indexing.

Minimum time----------4 word times
Overflow----------Not a factor
Branch Control-------Not affected
Registers affected---INDEX Register and/or either or both Accumulators

DIVIDE UPPER **DVU**

INSTRUCTION WORD FORMAT

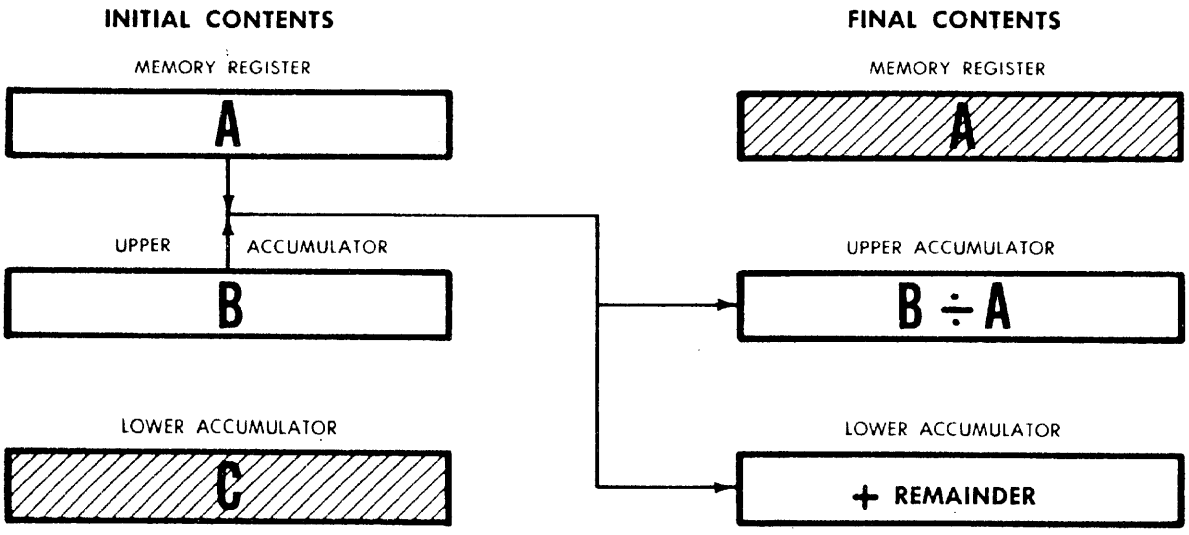
COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
10	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 1010 4	5	11 12 17	18	24 25 30	31

The contents of the UPPER Accumulator are divided by the contents of the memory location specified by the Data-address. The quotient is left in the UPPER Accumulator, and the remainder is left in the LOWER Accumulator. The remainder is always a positive value, regardless of the sign of the quotient, so that the dividend minus the remainder will always equal the divisor times the quotient.

The Data-address may be modified by indexing.

NOTE: This instruction should not be executed in Repeat Mode nor with the eight-word LOWER Accumulator.

Minimum time---------70 word times
Overflow-------------Turns on Branch Control
Branch Control-------Turned on by overflow
Registers affected---UPPER and LOWER Accumulators



DIV DIVIDE

INSTRUCTION WORD FORMAT

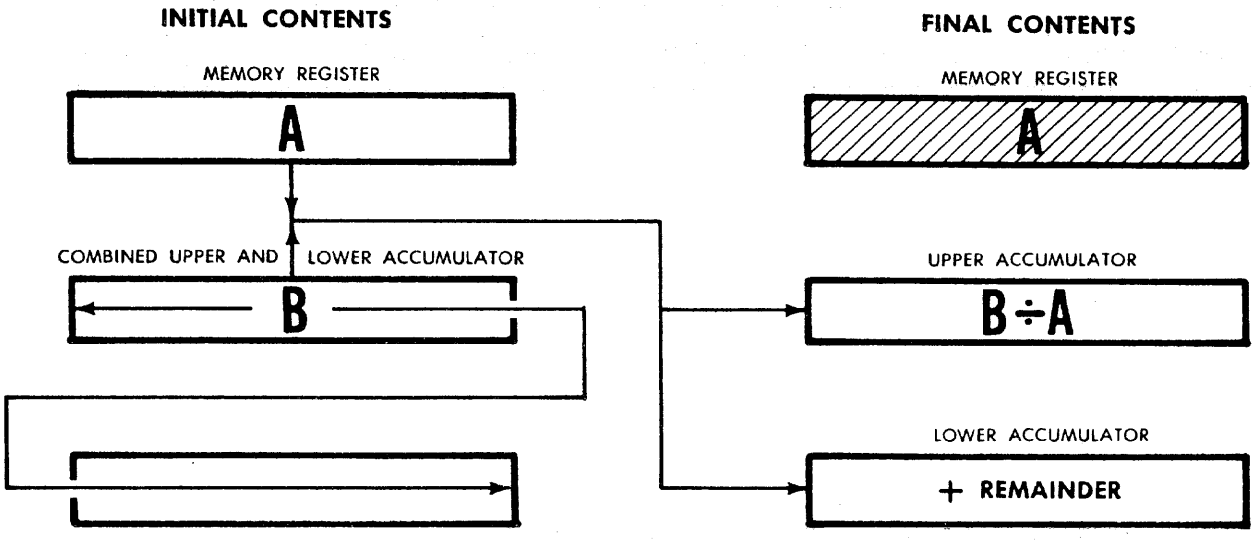
COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
11	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 1011 4	5	11 12 17	18	24 25 30	31

The contents of the double length accumulator (combined UPPER and LOWER) are divided by the contents of the memory location specified by the Data-address. The quotient is left in the UPPER Accumulator, and the remainder is left in the LOWER Accumulator. The remainder is always a positive value, regardless of the sign of the quotient, so that the dividend minus the remainder will always equal the divisor times the quotient.

The Data-address may be modified by indexing.

NOTE: This instruction should not be executed in Repeat Mode nor with the eight-word LOWER Accumulator.

Minimum time----------70 word times
Overflow--------------Turns on Branch Control
Branch Control--------Turned on by overflow
Registers affected---UPPER and LOWER Accumulators



SHIFT RIGHT OR LEFT **SRL**

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
12	000 OR 001	(SHIFT COUNT)	(NEXT ADDRESS)		0/1
0 1400 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000, shifts the contents of the double length accumulator (combined UPPER and LOWER) to the right by the number of bit positions specified in the Shift Count. Bits shifted out of the low order bit position are lost. The sign bit is duplicated in vacated bit positions.

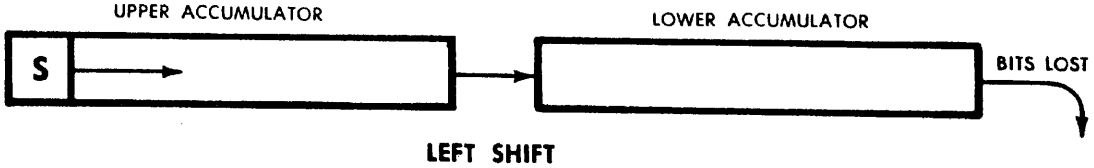
If the D-track field contains 001, shifts the contents of the double length accumulator to the left by the number of bit positions specified in the Shift Count. If overflow occurs, turns on Branch Control. Bits shifted out of the sign position are lost. Vacated bit positions are filled with zeroes.

The Shift Count and/or direction of shift may be modified by indexing.

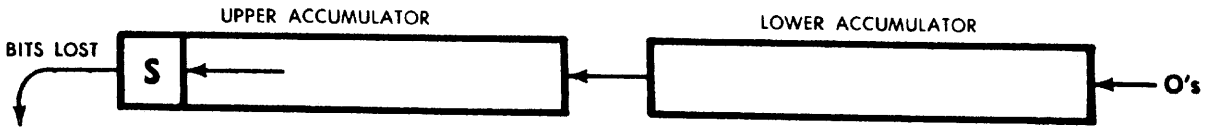
NOTE: This instruction should not be executed in Repeat Mode nor with the eight-word LOWER Accumulator.

Minimum time----------7 word times plus 1 for each bit position shifted.
Overflow--------------Turns on Branch Control.
Branch Control--------Turned on by overflow
Registers affected---UPPER and LOWER Accumulators

RIGHT SHIFT



LEFT SHIFT



SLC

SHIFT LEFT AND COUNT

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
13			(NEXT ADDRESS)		0/1
0 1101 4	5	11 12	17 18	24 25	30 31

The value contained in the double length accumulator (combined UPPER and LOWER) is shifted to the left until bit 1 contains the first significant magnitude bit, or until the sum of the D-sector value of the instruction word and the number of bit positions shifted equals 64. Following the shift, the LOWER Accumulator is cleared to zero, and the sum of the D-sector value and the number of shifts is placed in LOWER Accumulator bits 12 through 17, modulo 64. That is, a sum of 64 will appear in the LOWER as zero.

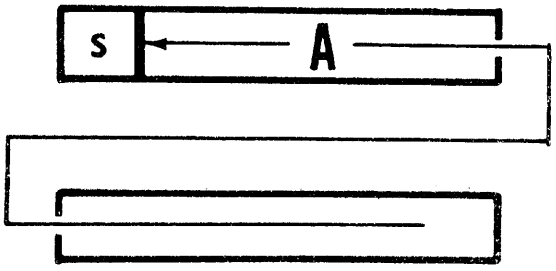
The Data-address may be modified by indexing.

NOTE: This instruction should not be executed in Repeat Mode nor with the eight-word LOWER Accumulator.

Minimum time---------7 word times plus 1 for each bit position shifted.
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---UPPER and LOWER Accumulators

INITIAL CONTENTS

COMBINED UPPER AND LOWER ACCUMULATOR



FINAL CONTENTS

UPPER ACCUMULATOR

LOWER ACCUMULATOR

MULTIPLY **MPY**

INSTRUCTION WORD FORMAT

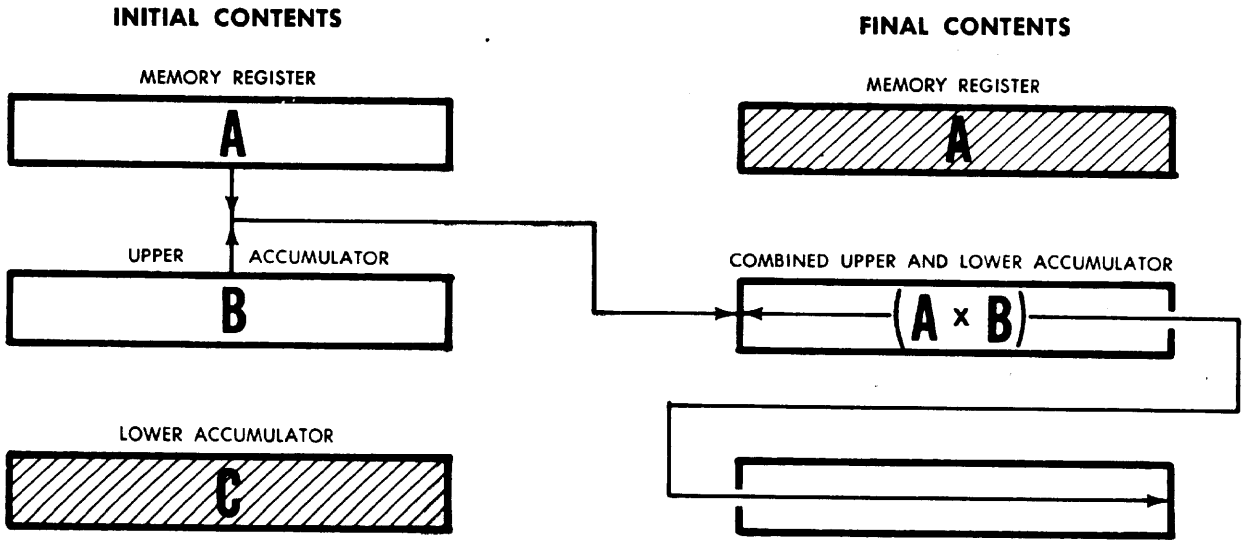
COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
14	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 1 1 1 0 4	5	11 12 17	18	24 25 30	31

The contents of the UPPER Accumulator are multiplied by the contents of the memory location specified by the Data-address. The resulting double length product is held in the combined UPPER and LOWER Accumulators.

The Data-address may be modified by indexing.

NOTE: This instruction should not be executed in Repeat Mode nor with the eight-word LOWER Accumulator.

Minimum time----------70 word times
Overflow--------------Not a factor
Branch Control--------Not affected
Registers affected---UPPER and LOWER Accumulators



MPT

MULTIPLY BY TEN

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
15	000 OR 064	(ANY)	(NEXT ADDRESS)		0/1
0 1 1 1 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000, multiplies the contents of the UPPER Accumulator by 10_{10}, retaining the product in the UPPER Accumulator.

If the D-track field of the Instruction Word contains 064, multiplies the contents of the LOWER Accumulator by 10_{10}, retaining the product on the LOWER Accumulator.

The D-track value may be modified by indexing.

NOTE: Unlike MPY, this is an integral multiply in which the "q" of the product in the affected accumulator is the same as the "q" of the multiplicand. The sign bit is treated as another magnitude bit.

Minimum time---------4 word times
Overflow-------------Does not turn on Branch Control
Branch Control-------Not affected
Registers affected---UPPER or LOWER Accumulator

PRINT FROM D-ADDRESS **PRD**

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
16	000 ⟶ 127	(ANY)	(NEXT ADDRESS)		0/1
0 / 0000 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000⟶063, prints the character it represents on the selected output device(s). If the D-track field contains 064⟶127, selects the input and/or output devices or modes it represents. (See selection table below). The I/O interlock will be turned on, preventing the execution of another print instruction until completion of the current one, if the D-sector field contains any sector other than the first optimum sector (instruction location + 2). If this sector is specified, the interlock will be bypassed.

The D-track and D-sector values may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---None

INPUT/OUTPUT SELECTION CODES

D-track		
64	4500	Reader input
65	4500	Reader input--Punch output
66	4500	Reader input--Typewriter output
67	4500	Reader input--Punch and Typewriter output
68	4500	Typewriter input
69	4500	Typewriter input--Punch output
70	4500	Typewriter input--Typewriter output
71	4500	Typewriter input--Punch and Typewriter output
72	4410	Photo-reader, Forward and Search
73	4410	Photo-reader, Reverse and Search
74	4410	Photo-reader, Forward
75	4410	Photo-reader, Reverse
76-94		Available for additional units
95		Master reset (reset all units)
96		Available
97	4500	Punch output

98	4500	Typewriter output
99	4500	Punch and Typewriter output
100		Available
101	4500	Punch output
102	4500	Typewriter output
103	4500	Punch and Typewriter output
104,105		Search Mode
106	4440	High Speed Punch
107-124		Available
125		Copy Mode on
126		Copy Mode off
127		Reset output units

NOTES:

1. Selection of a new input device automatically resets the previous one. Only one input device may be in the system at a time.

2. Any combination of output devices may be included in the system at one time. A reset command is necessary to drop an output device from the system.

PRINT FROM UPPER **PRU**

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
17	000 ⟶ 127	(ANY)	(NEXT ADDRESS)		0/1
0 10001 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000⟶063, prints the character represented by the combination of bits 6 and 7 from the Instruction Word followed by the high order four bits of the UPPER Accumulator.

If the D-track field contains 064⟶127, prints the character represented by the high order six bits of the UPPER Accumulator.

The I/O interlock will be turned on, preventing the execution of another print instruction until completion of the current one, if the D-sector field contains any sector other than the first optimum sector (instruction location + 2). If this sector is specified, the interlock will be bypassed.

The D-track and D-sector values may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---None

EXT EXTRACT

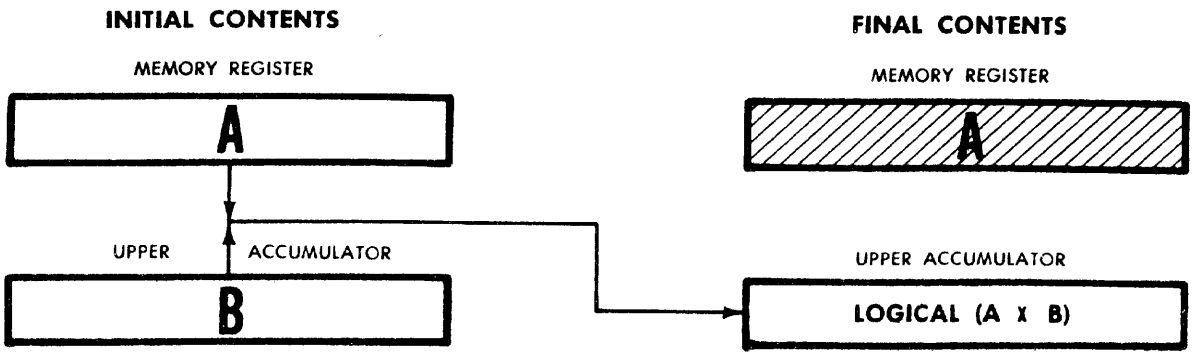
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
18	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 10010 4	5 11	12 17	18 24	25 30	31

Produces, in the UPPER Accumulator, the logical product of the contents of the UPPER Accumulator and the contents of the memory location specified by the Data-address. The resultant product will contain 1's in only those bit positions which are set to 1 in both the UPPER Accumulator and the memory word.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control--------Not affected
Registers affected---UPPER Accumulator



MASKED MERGE LOWER **MML**

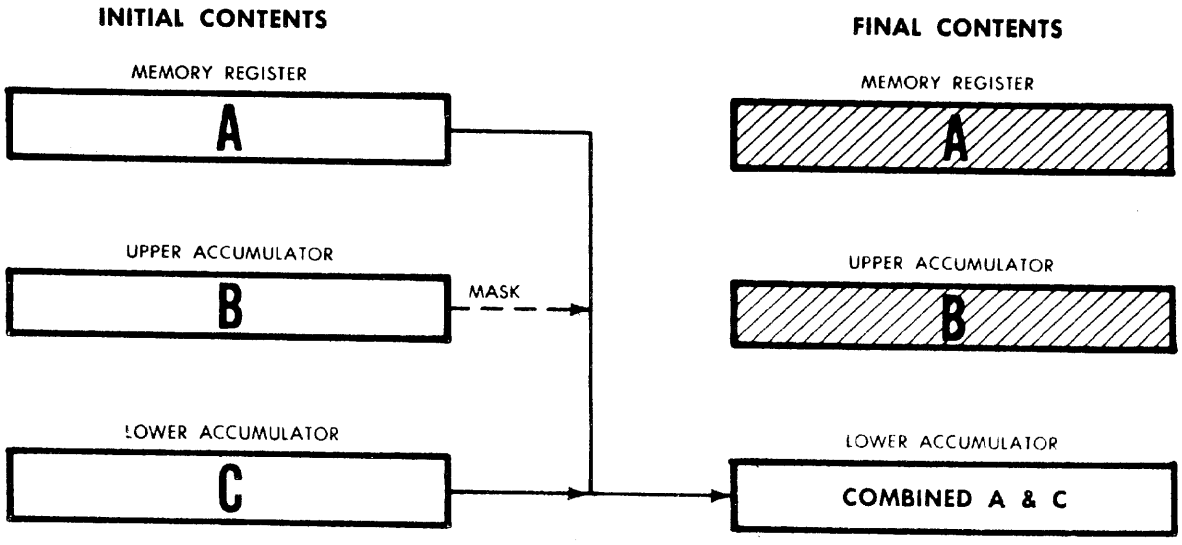
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
19	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 10011 4	5 11	12 17	18 24	25 30	31

The contents of the memory location specified by the Data-address are merged with the contents of the LOWER Accumulator under control of the mask in the UPPER Accumulator. In those bit positions where the UPPER Accumulator contains 0's, the LOWER Accumulator is retained. In those bit positions where the UPPER Accumulator contains 1's, the contents of the memory word replace the corresponding contents of the LOWER Accumulator.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control--------Not affected
Registers affected---LOWER Accumulator



CME

COMPARE MEMORY EQUAL

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
20	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
010100 4	5 11	12 17	18 24	25 30	31

Prior to performing the specified comparison, the Branch Control is turned off.

Selected bits of the UPPER Accumulator are then compared with corresponding bits in the memory location specified by the Data-address. Only those bit positions indicated by 1's in the LOWER Accumulator mask are compared. If the selected values are identical, the Branch Control is turned on to indicate a successful comparison.

If the instruction is being executed in the Repeat Mode, the Sector Reference Timing Track is copied into bits 25 → 30 of the INDEX Register during each comparison until a successful comparison is made or until completion of the repeat function. A successful comparison inhibits any further copying of the timing track. The memory location containing the value which compares successfully may then be determined by reference to the INDEX Register.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow-------------Not a factor
Branch Control-------Conditionally set "On" or "Off"
Registers affected---INDEX Register if executed in Repeat Mode.

COMPARE MEMORY GREATER **CMG**

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
21	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 10101 4	5	11 12 17	18	24 25 30	31

Prior to performing the specified comparison, the Branch Control is turned off.

Selected bits of the UPPER Accumulator are then compared with corresponding bits in the memory location specified by the Data-address. Only those bit positions indicated by 1's in the LOWER Accumulator mask are compared. If the selected value in memory is greater than, or equal to the selected UPPER Accumulator value, the Branch Control is turned on.

If the instruction is being executed in the Repeat Mode, the Sector Reference Timing Track is copied into bits 25 → 30 of the INDEX Register during each comparison until a memory value equal to, or greater than the accumulator value is found, or until completion of the repeat function. If such a value is found, any further copying of the timing track is inhibited. The memory location which contains this value can then be determined by reference to the INDEX Register.

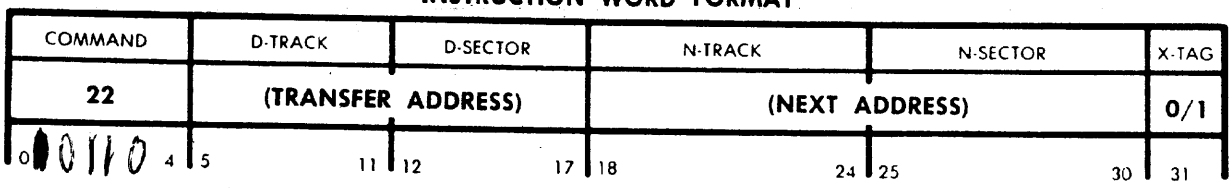
The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control--------Conditionally set "On or Off"
Registers affected---INDEX Register if executed in Repeat Mode.

TMI

TRANSFER ON MINUS

INSTRUCTION WORD FORMAT



If the UPPER Accumulator is negative (a "1" in bit position zero), control is transferred to the instruction specified by the Transfer Address. If the value is positive, the instruction has no effect, and the next instruction is that specified in the Next-address field.

The Transfer Address may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---None

TRANSFER ON BRANCH CONTROL **TBC**

INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
23	(TRANSFER ADDRESS)		(NEXT ADDRESS)		0/1
0 10111 4	5 11	12 17	18 24	25 30	31

If the Branch Control is on, control is transferred to the instruction specified by the Transfer Address, and the Branch Control is turned off. If the Branch Control is off, the instruction has no effect, and the next instruction is that specified in the Next-address field.

The Transfer Address may be modified by indexing.

Minimum time----------4 word times
Overflow----------Not a factor
Branch Control-------Set to "Off"
Registers affected---None

STU

STORE UPPER

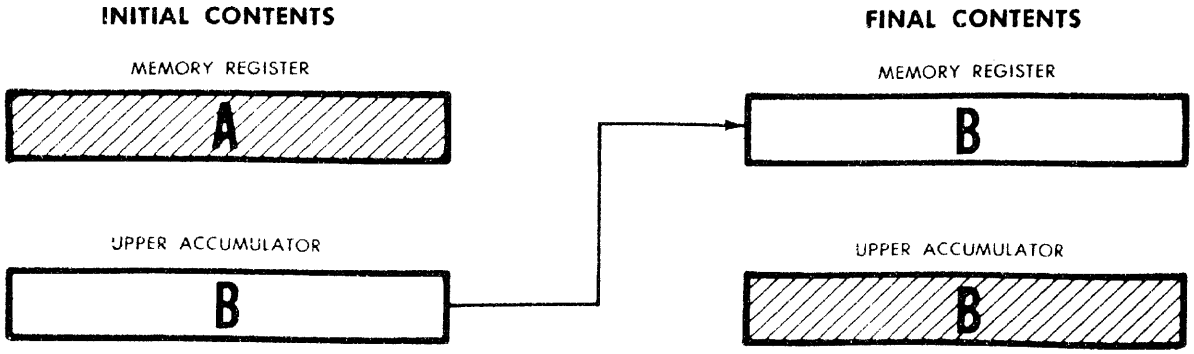
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
24	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
011000 0 4	5 11	12 17	18 24	25 30	31

Stores the contents of the UPPER Accumulator into the memory location specified by the Data-address, replacing its current contents.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control--------Not affected
Registers affected---Specified memory location



STORE LOWER **STL**

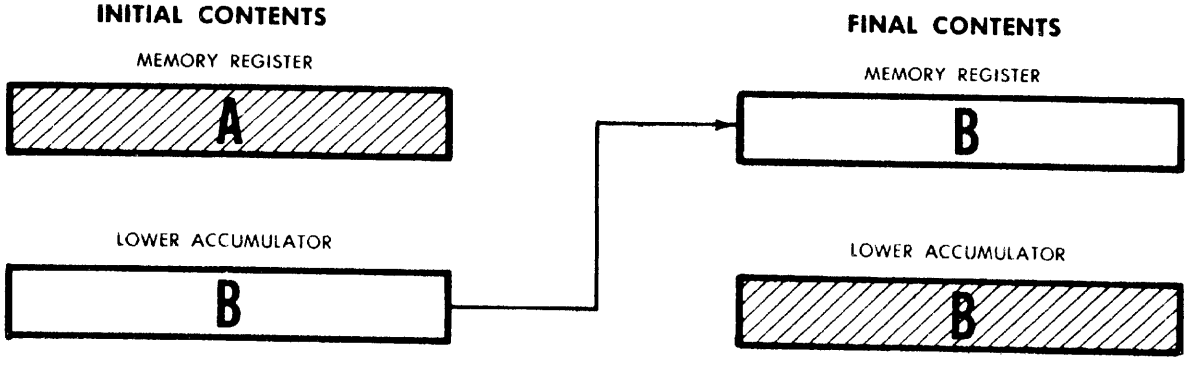
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
25	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 11001 4	5	11 12 17	18	24 25 30	31

Stores the contents of the LOWER Accumulator into the memory location specified by the Data-address, replacing its current contents.

The Data-address may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---Specified memory location



CLU

CLEAR UPPER

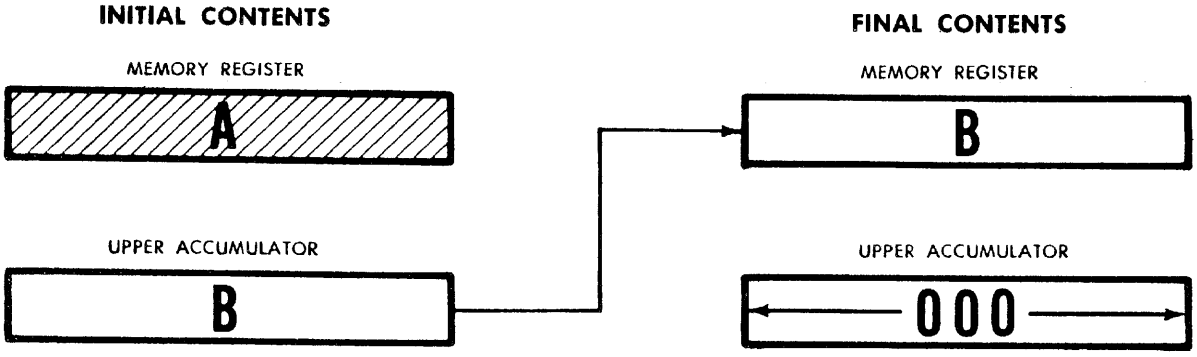
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
26	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0/1010 4	5 11	12 17	18 24	25 30	31

Stores the contents of the UPPER Accumulator into the memory location specified by the Data-address, replacing its current contents. Clears the UPPER Accumulator to zero.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Not a factor
Branch Control-------Not affected
Registers affected---Specified memory location and
UPPER Accumulator



CLEAR LOWER **CLL**

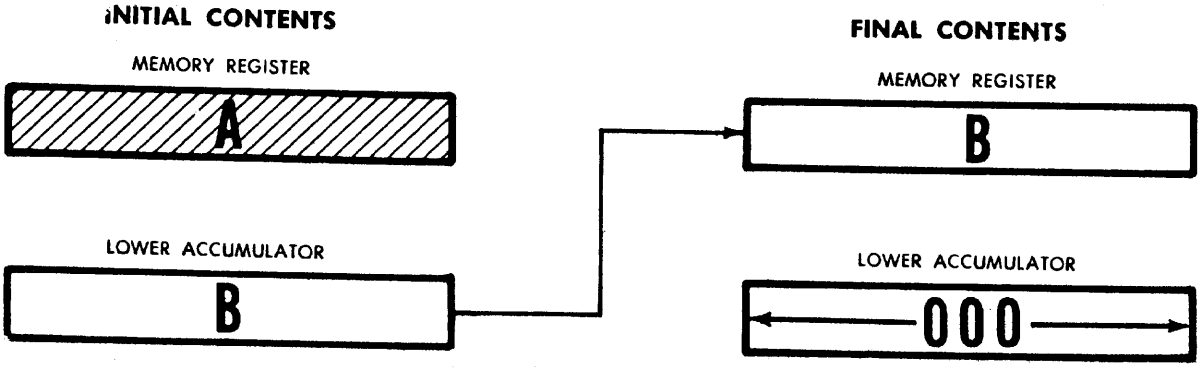
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
27	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 11011 4	5	11 12 17	18	24 25 30	31

Stores the contents of the LOWER Accumulator into the memory location specified by the Data-address, replacing its current contents. Clears the LOWER Accumulator to zero.

The Data-address may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Not a factor
Branch Control-------Not affected
Registers affected---Specified memory location and LOWER Accumulator



ADU

ADD TO UPPER

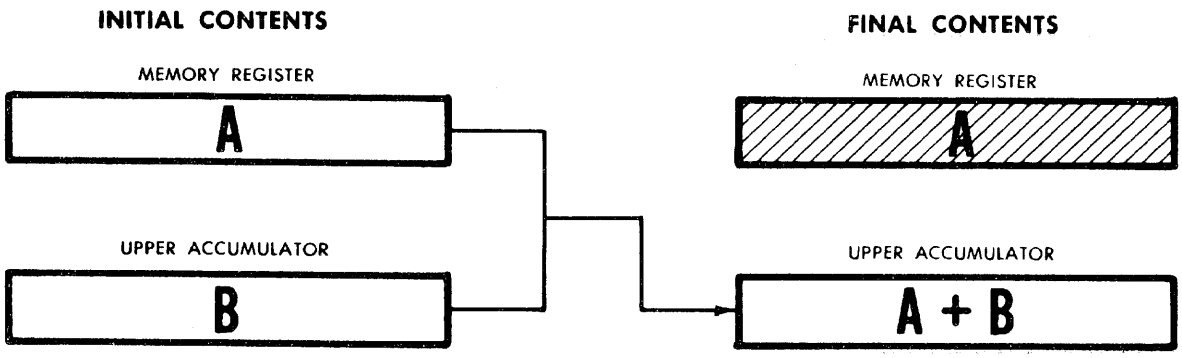
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
28	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 11100 4	5 11	12 17	18 24	25 30	31

Adds the contents of the memory location specified by the Data-address to the contents of the UPPER Accumulator. Retains the sum in the UPPER Accumulator.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Turns on Branch Control
Branch Control--------Turned on by overflow
Registers affected---UPPER Accumulator



ADD TO LOWER **ADL**

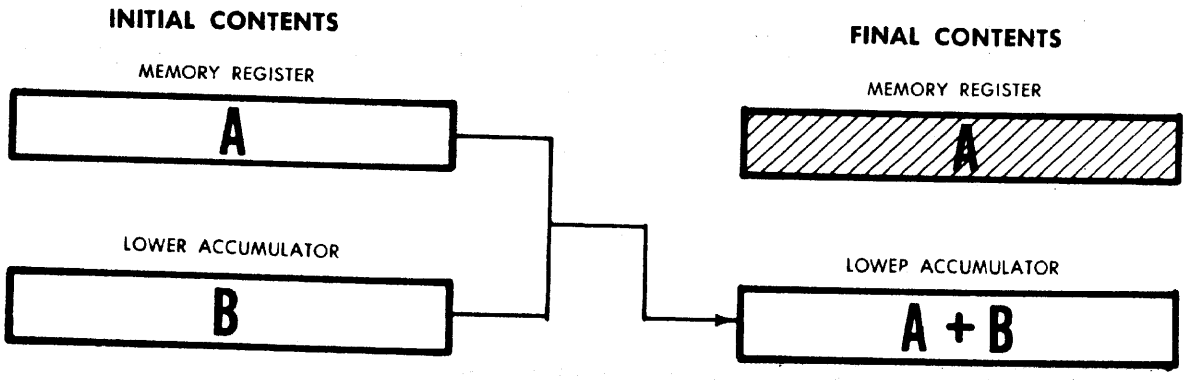
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
29	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 11101 4	5	11 12	17 18	24 25	30 31

Adds the contents of the memory location specified by the Data-address to the contents of the LOWER Accumulator. Retains the sum in the LOWER Accumulator.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow-------------Turns on Branch Control
Branch Control-------Turned on by overflow
Registers affected---LOWER Accumulator



SBU

SUBTRACT FROM UPPER

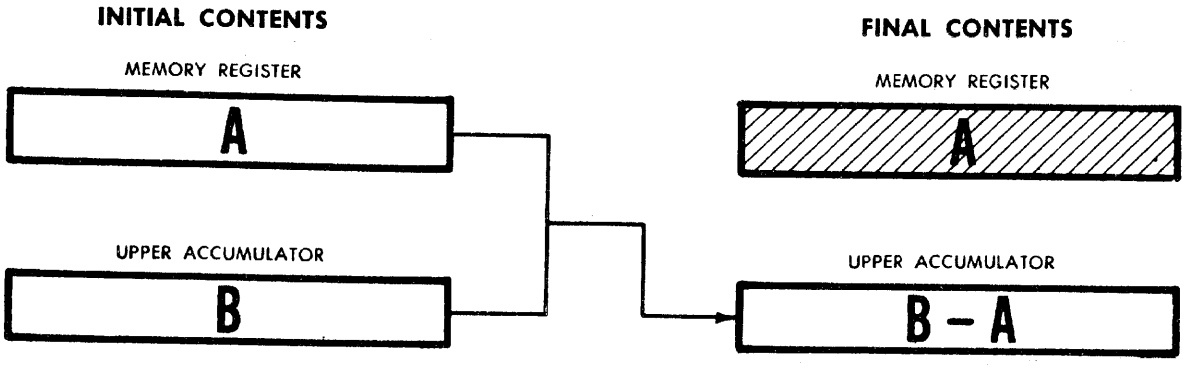
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
30	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 11110 4	5 11	12 17	18 24	25 30	31

Subtracts the contents of the memory location specified by the Data-address from the contents of the UPPER Accumulator. Retains the difference in the UPPER Accumulator.

The Data-address may be modified by indexing.

Minimum time---------4 word times
Overflow-------------Turns on Branch Control
Branch Control-------Turned on by overflow
Registers affected---UPPER Accumulator



SUBTRACT FROM LOWER **SBL**

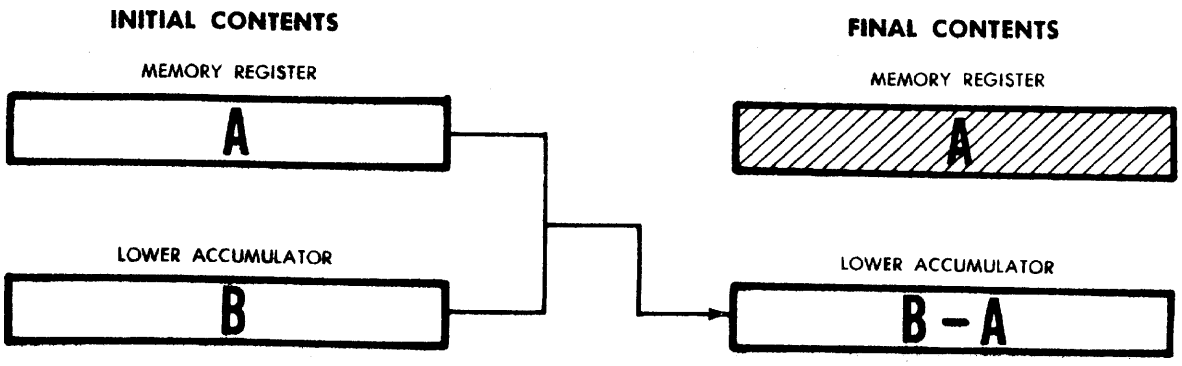
INSTRUCTION WORD FORMAT

COMMAND	D-TRACK	D-SECTOR	N-TRACK	N-SECTOR	X-TAG
31	(DATA ADDRESS)		(NEXT ADDRESS)		0/1
0 11111 4	5 11	12 17	18 24	25 30	31

Subtracts the contents of the memory location specified by the Data-address from the contents of the LOWER Accumulator. Retains the difference in the LOWER Accumulator.

The Data-address may be modified by indexing.

Minimum time----------4 word times
Overflow--------------Turns on Branch Control
Branch Control--------Turned on by overflow
Registers affected---LOWER Accumulator



5 PROGRAMMING TECHNIQUES

Programming, in general, may be thought of as encompassing three separate areas of activity--organization, coding and testing.

Program organization involves a determination of the processing methods to be used, the format of input and output data, the storage and arrangement of computational tables and constants, and the sequence in which the various operations within the program are to be performed. It is often convenient to divide a lengthy program into a number of smaller units, called subroutines, each of which performs some well-defined function necessary to the operation of the program.

Program coding is the process of perparing the step-by-step list of instructions that the computer is to execute in performing a required processing task. This coding may be in absolute form (machine language) which is directly assimilable by the computer, but is more often in some symbolic form which, in turn, requires translation by a previously written program as it is loaded into the computer. In any case, the coded program must represent an unambiguous statement of the operations to be performed, the locations of the data to be operated upon, and the sequence of execution of the included instructions.

Program testing, or "debugging", necessarily follows the coding of a program or subroutine. It is a rare program which functions perfectly on the first try. Normally, the first consideration in debugging is the logical design of the program; that is, the sequencing and storage of the program and its data. When it has been determined that the program has no unexplained halts or endless loops, it is usual to verify some program output against predetermined expected results.

This being accomplished, it is advisable to test the limits of the program with respect to the designed value ranges and data handling capacity. Finally, if the program has been written and checked out in the form of individual subroutines, an assembly test must be run to eliminate any storage conflicts and to assure proper communication between these subroutines.

PROGRAM ORGANIZATION

In order to effectively organize a program to be executed by the computer, the problem must first be defined in such a way as to enable the programmer to evaluate the program requirements in terms of input, output, table structure and memory utilization.

Consider the following hypothetical problem:

> An air freighter wishes to provide the fastest possible service to any destination from any one of five distribution bases. He has at his disposal 25 aircraft consisting of five each of five different types. Each type has a different cruising speed and range. Following each delivery, the aircraft is to be routed to that base which is nearest in terms of time. An aircraft may be removed from service or returned to service at any time. When an aircraft has logged 100 hours of flight time, it must be removed from service for maintenance.
>
> At any given time, the aircraft may be dispersed in any manner among the five bases or in the air. Following any landing, the aircraft must be allowed one hour of ground time before a subsequent take-off.

The program must accomplish the following functions:

1. In response to an input message, select the aircraft which will provide the fastest service to the specified destination. The input message is to consist of the name of the destination, the time of day, and the course, distance and wind from each of the five bases.

2. Print an Aircraft Assignment message consisting of departure base, destination, ETA to destination, new assigned base, ETA to new base, and aircraft number.

3. Print a note if service cannot be provided to a specified destination.

4. Print a note if more than 12 aircraft are out of service.

5. In response to an input message, remove a specified aircraft from service or restore an aircraft to service.

INPUT MESSAGES

Two types of input message must be recognized and processed by the program; an Aircraft Assignment Request message, and an Aircraft Availability message. The format of the input message should be such that it imposes no undue inconvenience on the person preparing the message, and that it may be handled in logical sequence by the program.

The A/C Assignment Request message may be conveniently composed as follows:

```
Destination *
Time of Day*
Base * Course/Distance/ * Wind Direction/Wind Velocity/ *
Base * Course/Distance/ * Wind Direction/Wind Velocity/ *
  "      "       "          "       "      "      "
  "      "       "          "       "      "      "
  "      "       "          "       "      "      "
END *
```

Destination may be any place name not exceeding 42 characters in length, including spaces.

Time of Day is entered in hours and minutes of the 24 hour clock.

Base must consist of the city name only and must be one of the five departure points, Boston, Atlanta, Chicago, Dallas and San Francisco.

Course/Distance/ must be entered as a compass course of up to 3 digits followed by a slash (/) followed by a distance of up to 4 digits followed by a slash (/). Wind Direction/Wind Velocity/ must be entered as a compass direction from which the wind is blowing followed by a slash (/) followed by a velocity of up to 3 digits, followed by a slash (/). Wind Velocity may not exceed 200 knots. END is entered as the word "END" to mark the termination of the message.

The A/C Availability message must give the identification number of the aircraft involved, and indicate whether the aircraft is being made available or unavailable. The message may be set up in the following form:

AIRCRAFT NO. DD-L *

Position "L" will contain the single character U or A, standing for "unavailable" or "available", and must be the last character preceding the stop code. Position "DD" will contain a 2 digit aircraft number and must precede the stop code by 3 character positions.

OUTPUT MESSAGES

Three types of output message are required of the program; an Aircraft Assignment message, an Aircraft in Maintenance message, and a Service Availability message. For the purpose of our hypothetical problem, the A/C Assignment message will consist of the headings, DEPARTURE, DESTINATION, ETA, NEW BASE, ETA, and A/C NO., with the appropriate message entry beneath each heading. If an ETA falls on a day other than the day of the message, it will be followed by a plus sign (+) with a number indicating the number of days following the day of the message. Thus, an ETA of 0200 + 2 will mean 2:00 A.M. on the second day following the message.

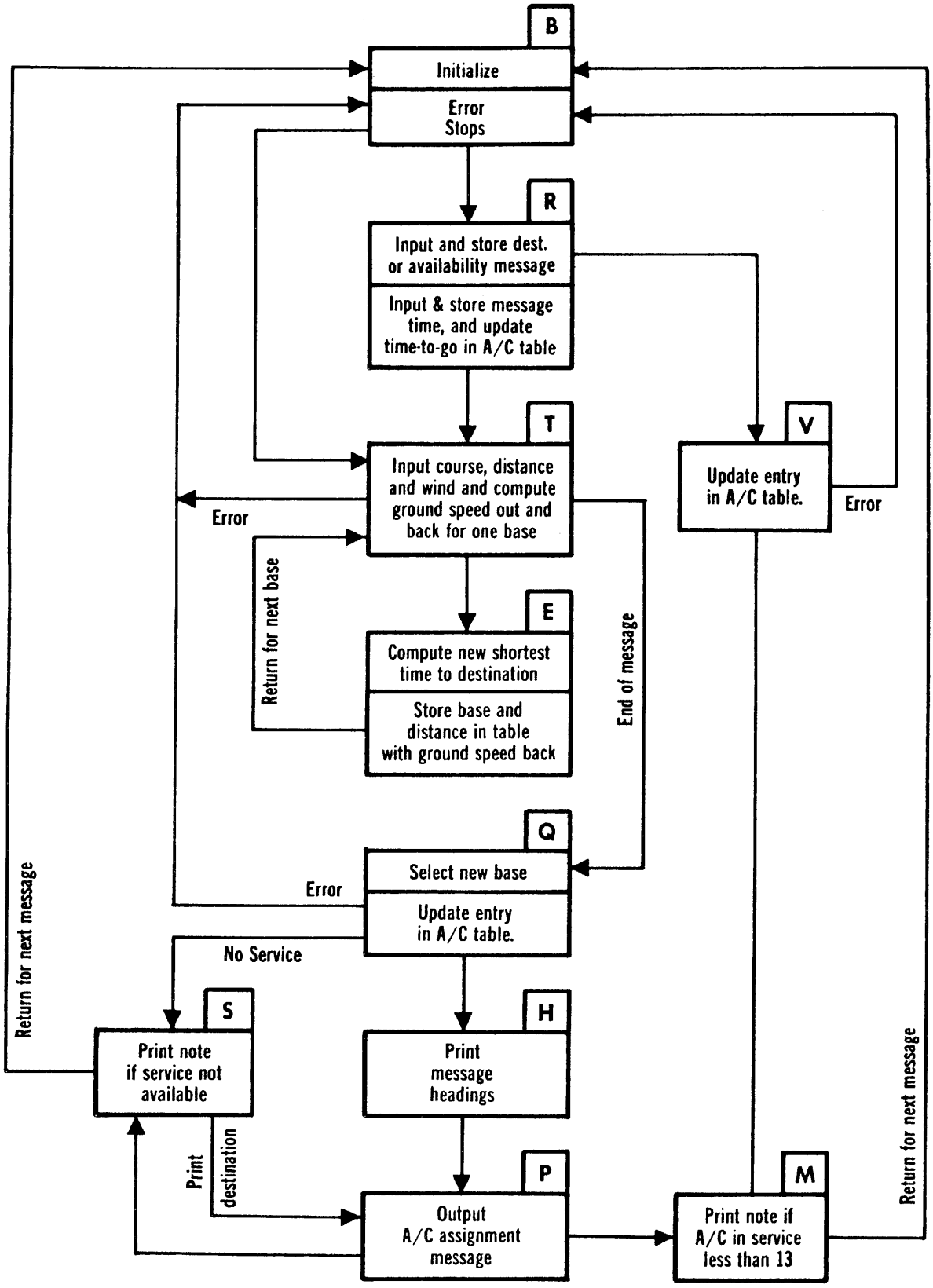


FIGURE 8. System Flow Chart For Example Problem

The Aircraft in Maintenance message will be printed whenever more than 12 aircraft are out of service and will read as follows:

DD AIRCRAFT IN MAINTENANCE

Position DD will contain the number of aircraft in maintenance. The Service Availability message will be printed when no aircraft of sufficient range are available for service to a requested destination and will read as follows:

SERVICE NOT AVAILABLE TO L....L

Position L ··· L will contain the requested destination.

SYSTEM FLOW CHART

Having defined the problem and its input and output requirements, it is usually helpful to prepare a system flow chart to indicate in broad terms what each subroutine of the program is to accomplish and to establish the lines of intercommunication between subroutines.

Such a flow chart for the problem under discussion identifies ten separate subroutines, each performing a portion of the overall problem solution and each communicating with one or more associated subroutines. Breaking a program down in this manner facilitates its preparation in that each subroutine can be separately coded and checked out before attempting to run the whole program.

TABLE ORGANIZATION AND STRUCTURE

Many programs will consist, in part, of data tables to be used or maintained by one or more of the subroutines incorporated in the program. The structure of the individual tables will be dictated by the manner in which they are to be used. It is not always possible to completely define the format of a table until the coding of the subroutine concerned is accomplished. However, the tables may be roughly defined and later modified as necessary to adapt them to the coding logic.

For the program under discussion, the primary table is an Aircraft Status Table consisting of 25 words, each word containing six items of information relating to the availability status of a particular aircraft. The word format for the table is as follows:

S	LOGGED TIME	A/C No.	TYPE	BASE	TIME-TO-GO
0	1 7	8 12	13 15	16 18	19 31

Bit 0 - Status indicator. 0 = In service. 1 = In maintenance.
Bits 1 - 7 -- Air time logged since maintenance in hours (0 -- 100)
Bits 8 - 12 -- Aircraft number (1 -- 25)
Bits 13 - 15 -- Aircraft type (0 -- 4)
Bits 16 - 18 -- Current base assignment (1 -- 5)
Bits 19 - 31 -- Time in minutes until a dispatched aircraft is available for re-assignment.

There are two smaller tables which are directly associated with the Aircraft Status Table as follows:

Airspeed Table -- five words containing the cruising speed in knots at a "q" of 31 for each of five aircraft types.

Range Table -- five words containing the range in minutes at a "q" of 31 for each of five aircraft types.

When an A/C Assignment Request message is received, the departure base is input in four-bit mode and compared with the contents of a six word Base Table in order to determine the identification number of the base. The last word of the Base Table contains the four-bit configuration for END and serves to identify the end of a message.

For solution of the wind triangle to compute ground speed, a 91 word table of sines is used with the values scaled at a "q" of 1.

Four tables are used to hold preset print images for the typed program output. In addition to the above preset tables, the program generates temporary tables as follows:

Destination Storage -- eight words used to hold the input destination in alphanumeric form until it is printed as part of an output message.

Sin W Storage -- Five words used to hold the sine of the angle opposite the wind vector for five airspeeds.

Ground Speed to Destination -- Five words used to hold the computed ground speeds to destination for five aircraft types.

Ground Speed to New Base -- 25 words used to hold ground speeds and distances for five aircraft types to five bases.

PROGRAM CODING

A program may be coded in any one of several forms or languages each with its own rules of procedure to be followed. As previously stated, machine language implies numerical coding in a form which corresponds directly with the internal word and memory formats employed in the design of the computer. This language, although meaningful to the computer, requires considerable experience on the part of the programmer before it can be handled with any degree of facility. It further imposes on the programmer the burdens of optimizing and memory storage allocation.

Compiler languages are problem oriented languages which require processing by a compiler program for conversion to a language usable by the computer or by an assembly program. Compiler languages normally are in a form which closely approximates standard algebraic notation. The RPC-4000 compiler, COMPACT, is the subject of a separate publication.

The language which will best serve our purpose in discussing programming techniques is the language of the assembly program, ROAR (RPC-4000 Optimizer and Assembly Routine). An assembly program permits coding in a symbolic language in which each machine instruction is represented by one symbolically noted assembly instruction. The assembly program ROAR, will interpret mnemonic commands and will assign optimum absolute memory locations for locations which are expressed symbolically. It will also act upon a number of pseudo-instructions to reserve blocks of memory, set up addressable regions for block storage, set up constants, etc. These pseudo-instructions will be discussed as they are encountered in out study of techniques.

THE ROAR CODING FORMAT

The coding sheets used with ROAR consist of five fields with the headings, Location, Order, Data-address, Next-address and Comments. The vertical line to the right of each field implies a "stop" code (*) which must be entered into the punched symbolic input tape. ROAR requires five such "stop" codes for each coding sheet entry, whether or not the associated field is blank.

The Location, Data-address and Next-address fields may contain an absolute memory location or a symbol representing a location to be assigned by ROAR. Symbolic addresses may not normally exceed five characters in length and must contain at least one non-numeric character. Six-character addresses are used only for certain special addressing functions. Fields may be left blank subject to certain rules to be defined later.

The Order field will contain a machine command or a ROAR pseudo-instruction and may be in mnemonic or numeric form. An indexed command is indicated by appending an X immediately preceding the order code. If a numeric command is used, the X must precede at least two digits.

The Comments field may contain any information the programmer desires except that no "stop" code may be used except the one which identifies the end of the field.

RPC-4000 CODING SHEET

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
START	RAU	1234		Bring contents of location 1234
	XSTU	SAVE	NEXT	Store in SAVE plus index value
NEXT	X24	HOLD		Store in HOLD plus index value
	HLT	0	CONT	Halt

SPACE RESERVATION

Usually, the first step in coding a program is to assign storage locations for various tables and word blocks required by the program. It is also often desirable to pre-establish absolute addresses for particular key symbols, as for example, the initial location of the program.

The ROAR pseudo-instruction, RES, will cause ROAR to reserve the block of memory between and including the specified initial and final location. By "reserve" is meant that ROAR will not assign, to any instruction or constant, any location in the reserved area except through the use of the pseudo-instruction, EQV.

The pseudo-instruction, REG, sets up and reserves a block of memory which may be addressed by a special region address symbol, containing a region tag followed by a number indicating the word location relative to the initial location of the region. The pseudo-instruction EQR, will equate a location symbol with any desired memory location and will then reserve this location, preventing its subsequent assignment to another symbol, except through the use of the pseudo-instruction EQV.

The pseudo-instruction, EQV, functions essentially the same as EQR, except that it is assumed that the location has been previously reserved.

Consider the following coding sequence for space reservation in our example problem:

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	RES	4000	12200	ROAR, CATRO, BOOTSTRAP
	REG	A03500	3531	Aircraft Status Table

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	REG	D03532	3539	Destination or Availability Message
	REG	T03540	3547	Elapsed time
	REG	W03548	3551	Instruction words
	REG	V03552	3556	Sin W for 5 airspeeds
	REG	B03600	3605	Base Reference Table
	REG	P03606	3610	Airspeeds
	REG	G03611	3615	Ground Speeds to Destination
	REG	R03616	3620	Range in minutes
	REG	M03621	3631	Base print image
	REG	N03632	3636	A/C Maintenance print image
	REG	C03637	3641	-Service Not Available- print image
	REG	H03642	3652	Column headings
	REG	S03700	3826	Sine Table
	RES	3827	3851	Ground Speeds to New Base
	EQV	EGSNB	3827	First location of Ground Speeds to New Base
	EQR	START	0	Beginning location of program

The first entry serves to confine the program to the lower 40 tracks of memory. Since ROAR, itself, does not occupy any memory below track 52, this allows us to keep ROAR intact in memory while checking out or operating our program. It also allows the space from track 40 through track 51 to be used for the bootstrap routine which ROAR provides for loading the assembled program into memory, and for utility programs, such as Change and Transfer which are helpful in debugging. The double access tracks, 123 thru 126 and the recirculating track, 127, are automatically reserved by ROAR and need not be reserved by the program.

The next 14 entries set up region storage for the tables discussed previously, and in one instance, Region W, for four program instructions which require contiguous memory storage. Region S, for example, will occupy 91 locations, beginning at location 3700 and ending at 3826. Location 3700 will be addressed in the program by the special regional address, A00001, and location 3826 by A00091.

The table of "ground speeds to new base" is not set up as a region, but the space is reserved by the RES pseudo-instruction. The initial location is symbolized EGSNB, hence the symbol is made equivalent to location 3827, using the pseudo-instruction EQV.

Finally, the beginning location of the program is established as START and is assigned the location 00000 which is reserved through the use of the EQR pseudo-instruction. Any reference in the program to the symbolic location START will be interpreted by ROAR as absolute location 00000.

Note that the location column in all of the above is left blank. Since none of the pseudo-instructions RES, REG, EQR, or EQV, generate any words to be stored in memory, a location for the pseudo-instruction is inap-

propriate. Note also that, in any field, leading zeroes may be omitted.

SUBROUTINE R

We will bypass, for the present, the discussion of initialization, since it cannot be determined what initialization is necessary until the other subroutines have been coded.

Subroutine R, as indicated on the System Flow Chart, is to input and store in memory the destination portion of an A/C Assignment Request message, or an entire A/C Availability message. It must, of course, determine which type of message is being processed.

If the message is an A/C Assignment Request, Subroutine R is to perform the additional functions of reading in, from tape or typewriter, the message time, and up-dating the Time-to-go item in the Aircraft Status Table.

It is advisable, for each subroutine in a program, to draw a detailed flow chart of the logical pattern to be followed in coding the subroutine. Such a flow chart enables the programmer to clearly visualize the instruction sequences necessary to accomplish the required functions. It is extremely helpful in avoiding the many pitfalls and logical traps which are a part of any complex problem.

The initial action of Subroutine R is to test the setting of SENSE SWITCH 1. If down, typewriter input and output will be selected: if not, tape reader input and typewriter output will be selected. It is assumed at this point that the LOWER Accumulator will have been set to 8-word length during initialization. An input is called for and stored in the eight word Region D. A test is then made to determine the type of message and, if an Aircraft Availability Message, control is transferred to Subroutine V. Otherwise, the time of day is input.

The time of day is input in hours and minutes of the 24 hour clock. The hours and minutes are now binarized separately, and converted to minutes in binary form. This time is then compared with the previous message time to compute an elapsed time and this elapsed time is duplicated in the eight words of Region T. The new time of day is then stored in location, MTIME.

A block of eight words is brought into the 8-word LOWER from the Aircraft Status table stored in Region A. The elapsed time is then subtracted from each of the eight words in the LOWER. An eight pass loop is programmed here in which each word in turn is exchanged into the UPPER Accumulator and tested for a negative sign bit. If negative, the associated aircraft has completed its last assignment and its time-to-go is set to zero. When eight words have been processed, they are precessed to their original positions in the LOWER, and stored back into Region A.

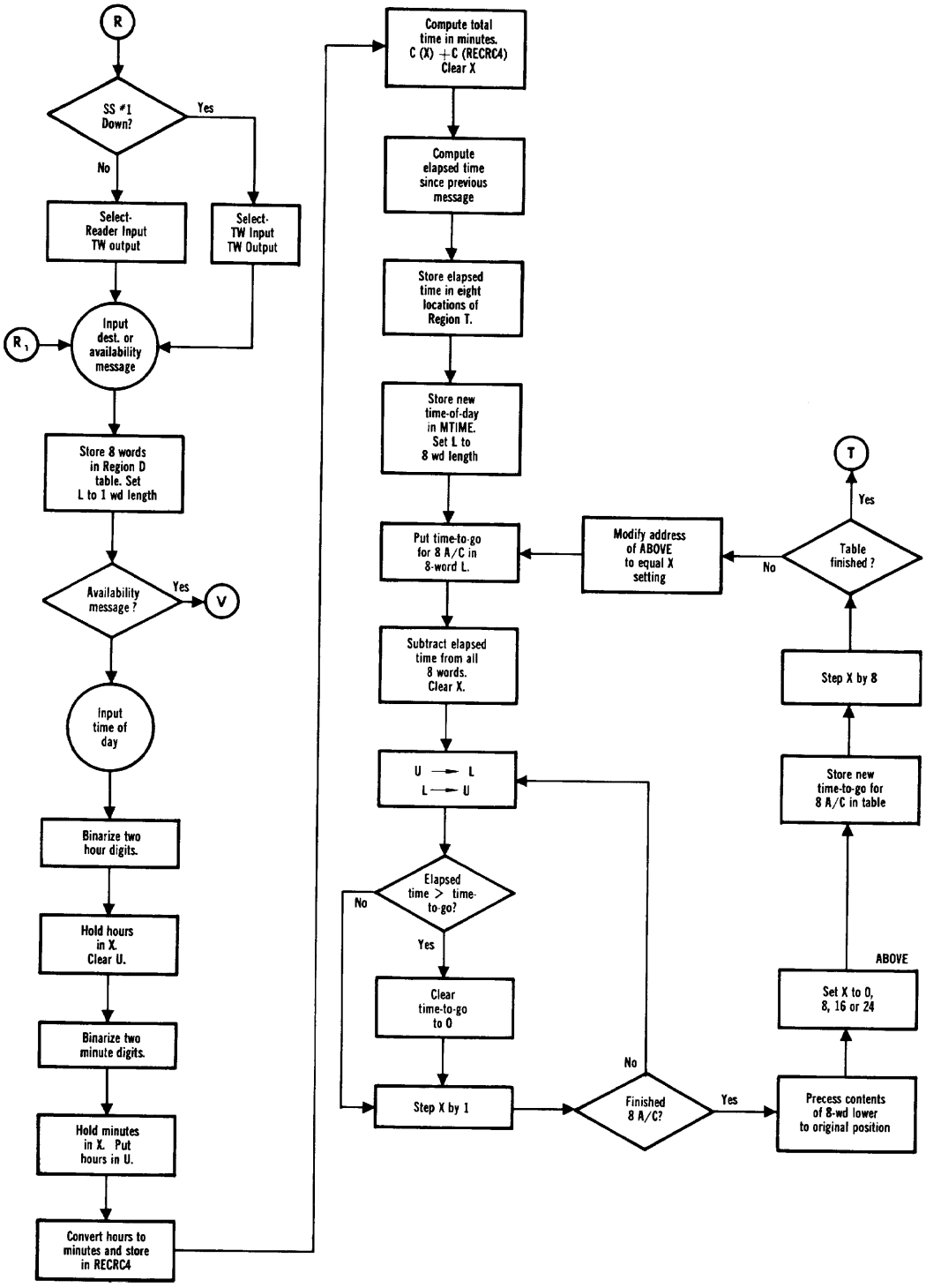


FIGURE 9. Flow Chart - Example Subroutine R

The index reference is increased by eight, and a test is made to determine whether the entire table has been updated. If not, another pass is made through the main loop to process the next eight words. If finished, control is transferred to Subroutine T.

Consider now the ROAR coding of Subroutine R.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	R		Message input
INPUT	SNS	100		Test Sense Switch 1
	TBC	TW		If down, go to TW
	PRD	6699	MREAD	Select reader input, typewriter output
TW	PRD	7099	MREAD	Select typewriter input, typewriter output
MREAD	INP	6400		Input destination or availability message
	LDC	NT7		
	CLL	D00001		Store input
	EXC	298		Clear U
	EXC	3698		Clear X, L = 1
	RAU	1A		
	RAL	3F		
	CMG	D00001		Check for number in bits 14-19
	TBC		AVMES	If yes, go to AVMES
	CLU	JUNK		Clear U
	EXC	598		Clear L and X
	INP	0		Time of Day
	EXC	298		L to U
	SRL	12		
	MPT	98		Binarize hours
	CLU	RECRCO		
	SRL	104	PR2	
3F	HEX	3	F000	Mask for last digit of A/C number
1A	HEX	1	A000	Test constant
NT7	DEC	24	7	Repeat count of 7
PR2	ADU	RECRCO		
	EXC	1298		Hours to X, 0 to U
	SRL	104		
	MPT	98		Binarize minutes
	CLU	RECRC7		
	SRL	104		
	ADU	RECRC7		
	EXC	1298		Minutes to X, hours to U
	SRL	102		Convert hours to minutes
	STU	RECRCO		
	SRL	104		
	SBU	RECRC0		

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	CLU	RECRC4		Store converted hours
	EXC	1298		Minutes to U, 0 to X
	ADU	RECRC4		Compute time in minutes
	EXC	198		Message time to L
	SBU	MTIME		Compute elapsed time
	TMI		POS	Test for minus
	ADU	DAY	POS	If yes, add 24 hours
POS	LDC	NT7		
	STU	T00001		Store elapsed time in 8 locations
	CLL	MTIME		Store new message time
	EXC	1698		L = 8
	RAU	TMSK	PR3	
DAY	DEC	31	1440	24 hours in minutes
PR3	LDC	NT7		
	XMML	A00001		Get current time-to-go for 8 A/C
	LDC	NT7		
	SBL	T00001		Compute new time-to-go
	LDX	0	FLIP	
FLIP	XEXC	390		U to L, L to U
	TMI		BYP	Test time-to-go for minus
	CLU	JUNK	BYP	If yes, clear to zero
BYP	XLDX	1		Step X
	CXE	9		Test for finish 8 A/C
	TBC		FLIP	If no, go to FLIP
	LDC	NT62		
	EXC	398	ABOVE	Precess L to initial position
ABOVE	LDX	00		
	LDC	NT7		
	XMST	A00001		Store new time-to-go's
	XLDX	8		Step X by 8
	CXE	32		Test for finish - full table
	TBC	TBELO		If yes, go to TBELO
	EXC	1298		X to U, U to X
	SAU	ABOVE		Set up for next pass
	EXC	1298	PR3	X to U, U to X
NT62	DEC	24	62	Repeat Count of 62
TMSK	HEX	0	1FFF	Mask for time-to-go

The first entry is the ROAR pseudo-instruction TAG. It requires no Location and no Next-address. It causes ROAR to append the character from the Data-address field to every symbolic location or address of four characters or less, as it is placed in ROAR's bookkeeping Symbol Table. This allows identical symbols to be used for other locations in the other subroutines of the program, without being ambiguous to ROAR in making its memory allocation assignments. Five character symbols are not tagged in this manner, and are used wherever communication

between subroutines is required.

There are two entry points in this subroutine, labelled INPUT and MREAD. The normal entry is INPUT. MREAD is used for entry only after certain error stops.

The rules governing blank fields are these. For any instruction word, either the Data-address or the Next-address, but not both, may be left blank. If one of these fields is left blank, the location field of the following instruction or constant must also be blank. If both the Data-address and the Next-address fields of an instruction are occupied, the Location field of the following instruction or constant must contain a symbolic or absolute location. When ROAR encounters a blank address, it supplies an optimum memory address and assumes that the following word is to be located at that address in memory. These rules do not apply to such pseudo-instructions as TAG where no memory allocations are involved.

The first action of Subroutine R is to select input and output devices depending on the setting of SENSE SWITCH 1. Note that the PRD instructions contain the artificial Data-sector 99. This causes ROAR to assign the same sector number as was assigned to the location of the instruction. This sector assignment has no bearing on optimization, but is necessary in order to honor the print interlock (See pages).

At location, MREAD, an Input instruction is executed to read into the 8-word LOWER a destination or an Aircraft Availability message. A Repeat Count of 7 is then set by the LDC instruction, and the eight words in the LOWER are cleared into Region D.

The UPPER Accumulator and the INDEX Register are now cleared to zero. The artificial sector number, 98, in the Exchange instructions causes ROAR to assign an optimum sector number for the Data-address.

At this point it is necessary to determine the type of message being processed. This is accomplished by testing the third from last character of the message just stored in Region D against a six-bit character, A, in the corresponding bit positions of location 1A. An A/C Availability message will always have a number in this character position. An Aircraft Assignment Request message will always have an alphabetic character or punctuation mark in this position and the magnitude of its bit pattern will always be equal to or greater than the character A. If the comparison test indicates that this character position contains a number, control is transferred to Subroutine V at location AVMES.

When it is determined that a destination is in Region D, the time of day is input with the LOWER in 1-word mode. This time must now be converted from hours and minutes, which are in the LOWER in BCD form, to minutes in binary form. The first character of the hours is placed in the low order four bits of the UPPER, multiplied by 10, and cleared into

the Recirculating Track, 127. The second character is now shifted into the UPPER and added to the value placed on Track 127. The hours in binary are then stored temporarily in the INDEX Register, and the minutes are binarized in the same manner. Next, the binarized minutes are stored in the INDEX Register while the hours are brought back to the UPPER and converted to minutes.

This could of course be accomplished by multiplying by 60. But for certain small values, it is faster to program this multiplication by a series of steps such as is used here. The hours are first shifted left two binary places, which produces a multiplication by 4. This value is stored in Track 127 and the hours, still in the UPPER, are shifted left four more places, producing a multiplication of the original value by 64. The word stored on Track 127 is now subtracted from the contents of the UPPER, leaving a value which is 60 times the original hours value. This is combined with the minutes held in the INDEX Register to complete the time conversion process.

Before proceeding with the next instruction sequence, there are several points which should be brought out. The artificial sector number 98 in the MPT instruction has the same function as was previously discussed in connection with the EXC instruction. References to Recirculating Track storage are coded in ROAR language with a 6-character symbol consisting of the letters RECRC followed by a prime modulo 8 sector number.

ROAR will choose as an absolute address the next optimum occurrence from Track 127 of a location with a sector number whose modulo 8 equivalent is equal to that entered in the symbol. Thus, for RECRC0, ROAR might substitute 12700, 12708, ---- 12756, whichever is most optimum.

The three locations, 3F, 1A, and NT7 all contain constants using ROAR pseudo-instructions, HEX or DEC. A HEX constant is entered with the first four characters in the Data-address field, and the last four characters in the Next-address field. Leading zeroes in a field may be omitted except that at least one character must be entered. The DEC constant is entered with the desired "q" in the Data-address field, and the decimal number to be converted in the Next-address field, including a decimal point, if appropriate, and preceded by a minus sign if negative.

Resuming our discussion where we were so rudely interrupted, we now proceed to compute the elapsed time since the last processed message. We subtract the previous message time held in location MTIME from our new time and test the sign of the result. If negative, we know that the previous message occurred yesterday, and we must add 24 hours to produce the true elapsed time. It is assumed here that less than 24 hours have elapsed between messages.

After completing the elapsed time computation, it is stored in all eight locations of Region T, and the new message time is stored in MTIME.

The LOWER Accumulator is now changed to 8-word mode and a mask defining

the time-to-go field of the Aircraft Status Table is placed in the UPPER. Location PR3 marks the beginning of the major or outside loop to be used in updating the Availability Table. The INDEX Register at this time remains set to zero from a previous instruction. The repeated and indexed MML instruction brings into the eight words of the LOWER, the contents of the time-to-go field from the first eight words of the Aircraft Status Table in Region A. The mask in the UPPER defines the bits to be brought from memory. The eight identical elapsed time values from Region T are subtracted from the eight time-to-go's with a single repeated SBU instruction.

Location FLIP marks the beginning of the inside loop in the updating sequence. The artificial sector number 90 directs ROAR to assign a sector equivalent to a modulo eight zero, so as to being L_0 into the UPPER on the first pass. L_0 is now tested for a negative sign bit. If minus, the elapsed time must have exceeded the time-to-go and the aircraft is now available for a new assignment. The time-to-go is, therefore, cleared to zero. Otherwise the new time-to-go has been computed and we go to location BYP where the INDEX Register is incremented by 1. Its setting is now tested to determine whether it has reached 9 and, if not, we return to location FLIP where the next LOWER Accumulator is brought into the UPPER at the same time that is replaced by the previously computed time-to-go.

We will exit from the minor loop when the Index value reaches nine, at which time eight time-to-go's will have been processed and each will have been shifted or precessed by one word position in the 8-word LOWER. The repeated EXC which follows the minor loop is necessary to precess the eight values to their original positions in the 8-word LOWER so that we may store them in their proper table locations. The INDEX Register is now set initially to zero and the new time-to-go's are stored into the first eight locations of Region A.

The INDEX Register is incremented by 8 and tested for a setting of 32. If not, the current setting is used to modify the Data-address of location ABOVE, so that on tne next pass the updated values will be stored into the proper 8-word block of Region A. We now return to PR3 to repeat the entire process for the next eight words from the table being processed. When the entire table has been updated and the INDEX Register setting has reached 32, we exit from Subroutine R to location TBELO, which is the initial location of Subroutine T.

SUBROUTINE T

Having updated the time-to-go field for all aircraft represented in the Aircraft Status Table, we may now proceed with the programming of the subroutine which will solve the wind triangle to compute ground speed.

As can be seen from the flow chart, the general procedure to be followed is this. A base is input as part of the A/C Assignment Request message, and checked against a table of bases. If it is not found, we exit to an

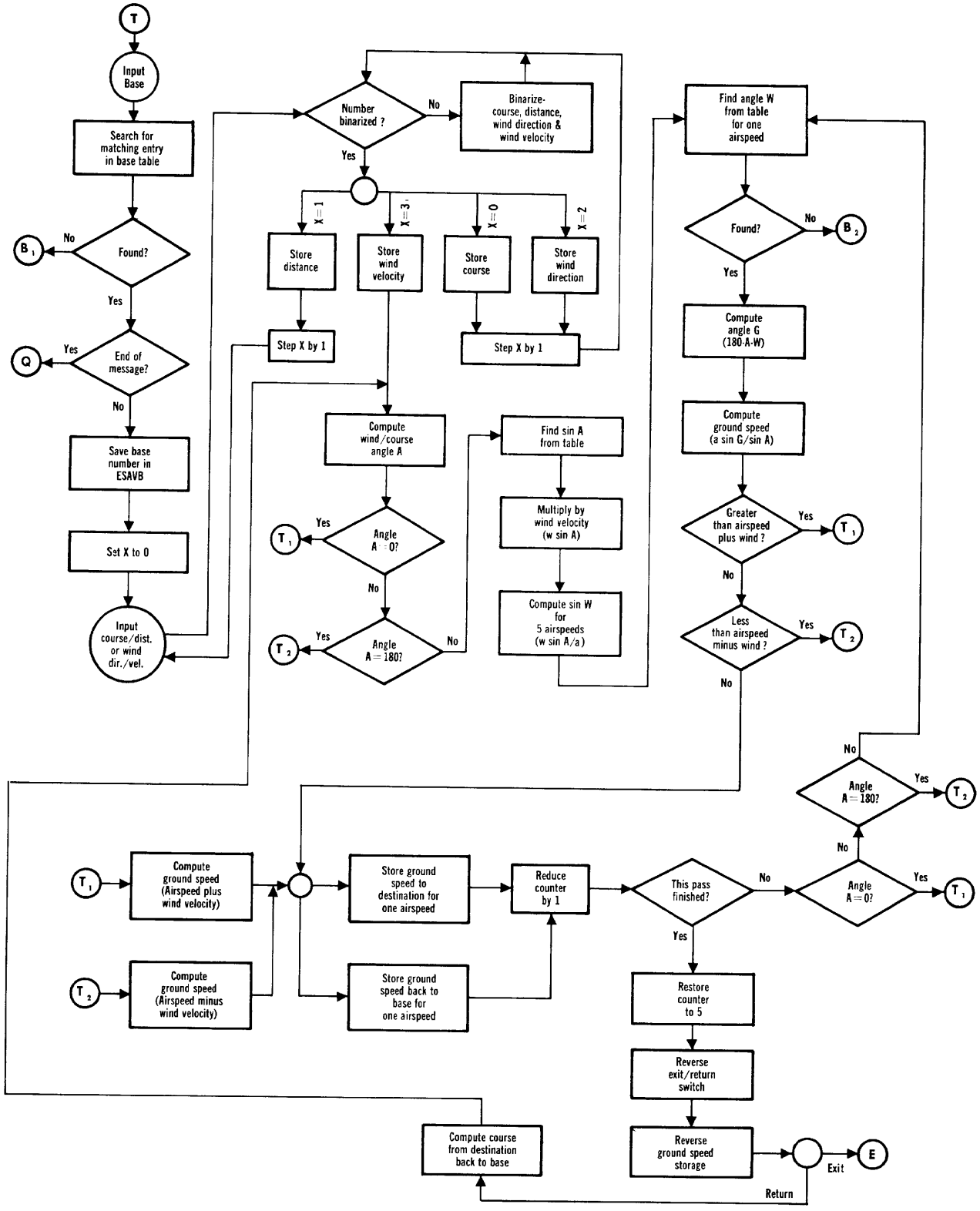


FIGURE 10. Flow Chart - Example Subroutine T

error halt in Subroutine B. If it is found to be an end-of-message code, we exit to Subroutine Q. If the base is found in the base table, its number is saved in location ESAVB, and its associated course, distance, wind direction and wind velocity are input, binarized and stored in turn.

The angle between course and wind is now computed and is tested for equality to 0 and 180. If either is true, we branch to a part of the subroutine which will handle these special cases. Otherwise, we find the sine of the angle from the table of sines in Region S. The sine of angle A is multiplied by the wind velocity and then divided by the airspeed for each of five aircraft types. We have now computed the sine of the angles between the airspeed and ground speed vectors by the equation, sin W = sin A / a.

At this point, we enter a five pass loop in which we will compute the ground speed to destination for each of the aircraft types. To accomplish this, we find angle W from the sine table, and compute angle G from the equation, G = 180 - A - W. We can then determine the ground speed from the equation, g = a sin G / sin A, where g represents the ground speed and G represents the angle between the airspeed and wind vectors. Having found the ground speed for one aircraft type, we store it in Region G and return for another pass through the loop until all five passes are complete.

We also wish to compute in this subroutine the ground speeds from the destination back to base. Since the procedure is the same as that just described, it is necessary only to change the ground speed storage location, set up the necessary exit switches, compute the new course, and loop back to repeat the wind triangle solution.

Coding for Subroutine T

The subroutine is entered at location TBELO, where a constant consisting of all 1-bits is brought into the UPPER Accumulator and duplicated in the LOWER through the EXC instruction. A base input is then called for in 4-bit mode. Although an alphabetic character cannot be completely defined in four bits, for our purpose the pattern produced will be sufficient to identify the base. In the case of SAN FRANCISCO, only the last eight characters entering the accumulator are of concern. UPPER and LOWER Accumulators are exchanged, putting a mask in the LOWER which is used with the repeated CME instruction to find a matching entry in the base table in Region B. If no match is found, we exit to an error halt in Subroutine B.

If a match is found, bits 25-30 of the INDEX Register will contain the sector number of the matching entry, plus 1. It is important for our purpose that Region B begin at sector 00. If INDEX Register bits 25-30 contain a 6, it indicates that the word END was read from tape. This

is determined by exchanging the INDEX Register contents into the UPPER and comparing these bits against the test constant at location THRU.

If it is an end-of-message, we exit to location NOMOR in Subroutine Q. Otherwise, we save the base reference number in ESAVB.

An input instruction is now executed, bringing into the accumulators the course and distance from base to destination. The termination of each of these values is marked by a slash (/) which has the numerical value 14 in 4-bit mode. The maximum number of characters which will be input is 9, which would bring no more than one into the UPPER. The course is binarized and, when the / is detected, we branch to the first location of Region W to store it in location CRSE. The INDEX Register is stepped by 1, and we return to BACK to binarize the distance. After storing the distance in EDIST, we loop back to PT2 to input the wind direction and velocity which are processed in the same manner. Note that we have used an indexed exit from an instruction sequence to one of four sequentially stored instruction words. In order to activate the XTBC, we precede it with an XCXE with a zero Data-address to unconditionally turn on the Branch Control.

The determination of angle A is fairly straightforward and should require no detailed explanation, but it should be pointed out that "course" refers to the direction from the departure base to the destination, while "wind direction" refers to the compass direction <u>from</u> which the wind is blowing.

Solving for sin W in the equation sin W = w sin A/a is simply a matter of using the first quadrant equivalent of angle A to index into the sine table to find sin A, and then multiplying by wind velocity and dividing by airspeed. This is done for an airspeed of 300 knots. To compute sin W for a 600 knot airspeed, we need only halve the first value by a right shift of 1. Now adding sin W for 300 knots, gives us sin W for 200 knots and halving this value, gives us sin W for 400 knots. Finally, multiplying by the constant .8, located in JUG, gives us sin W for 500 knots. These sin W values are stored in Region V in the order in which they are computed.

With sin W for one airspeed in the UPPER, we now search the sine table for the first entry equal to or greater than the contents of the UPPER. We get the found sector plus one from bits 25-30 of the INDEX Register, subtract one from it, and use it to set the INDEX Register bits 5-17 to the correct table reference value. Note here the ROAR symbology used to refer to the double access tracks. The first three characters denote the track. DB1 refers to Track 123, while DB2 refers to Track 125. The last three characters have the same significance as any other location symbol. Both DB1QIK and DB2QIK designate the same word on the drum, but the DB2QIK reference is 16 word times later than DB1QIK. Double access is used here to permit two references to the same word in 16 word times instead of the normal 64.

After setting the INDEX Register at location DB2QIK, the sine value which compared with sin W is brought from the sine table into the UPPER. The five instructions beginning with location PT5 compute an interpolation factor. If an overflow occurs as a result of the DIV instruction, it means that an exact match was found in the table and no interpolation is required. We, therefore, proceed to compute angle G from the equation, G = 180 - A - W. Since we are really interested in the first quadrant equivalent of angle G, we may simply use the sum of angle A and angle W, if this sum is 90 degrees or less.

If interpolation for angle G is required, we use the sequence of steps beginning at location NOVR to compute angle G. Actually, the nearest integral angle value above the true value is used as an index into the sine table to get the bracketing sine values, and the interpolation factor is applied to compute the sine of angle G.

At location STOW, the just computed sin G is stored in RECRCO. We then set the Repeat Count from the variable parameter in ENTRA. This parameter will vary from 4 to 0 depending upon the particular airspeed with which we are concerned. The repeated RAU instruction will bring into the UPPER each airspeed in turn from Region P, until the Repeat Count runs out. At this point, the required airspeed is in the UPPER Accumulator. The maximum and minimum possible ground speeds are now computed from the airspeed and wind velocity, and are placed in recirculating storage. Ground speed is now computed from the equation, g = a sin G/sin A. It is possible with a nearly direct head or tail wind to get a ground speed which lies outside the possible range. The computed ground speed must now be compared against these limiting values. If it exceeds the maximum value, we branch to TAIL. If it is less than the minimum value, we branch to HEAD. Otherwise, we go to PT7 where a Repeat Count is set up to enable us to store the ground speed in the proper location of Region G by a technique similar to that discussed above.

For those cases resulting in a branch to TAIL or HEAD, ground speed is computed simply by adding or subtracting wind velocity from airspeed before proceeding to PT7.

When a ground speed has been stored, we proceed to location DECR, where the Repeat Count parameter is reduced by 1 and tested for a negative value. If not negative, angle A is brought back into the UPPER. If it is 0, we branch to TAIL. If it is 180, we branch to HEAD. If neither, the Repeat Count is set up, the next sin W is brought into the UPPER, and we return to AGIN to compute the next ground speed.

When five ground speeds have been processed, the Repeat Count parameter will become negative at DECR plus 1, and at DECR plus 2, we will branch to location RSET. At RSET, ENTRA is again set to 4. The instruction at TERM is now modified to change its Next-address to GOON, and the original setting is saved in EXSW. The instruction at SAGS is modified to change its Data-address to EGSNB, which is the initial location of the table of ground speeds from destination to base. The original setting is saved in EGSSW. At GOON,

the course is brought into the UPPER, and the reciprocal course from destination back to base is computed. We now return to location STOB to compute and store the five ground speeds back to base.

When we again arrive at RSET, location ENTRA and the two switches at SAGS and TERM are reset to their original values, and at TERM control is transferred to location ENEXT in Subroutine E.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	T		Wind triangle solution for ground speeds
TBELO	RAU	ALLF		Set to all 1's
	EXC	3398		U to L, L = 1
	INP	0		Input A/C base
	EXC	398		Base to U, mask to L
	LDC	NT5		
	CME	B00001		Search base table
	TBC		ERRO1	If not there, go to ERRO1
	EXC	898		X to U
	RAL	SMSK		
	CME	THRU		Test for end of message
	TBC	NOMOR		Go to NOMOR if message complete
	CLU	ESAVB		Save base reference
	EXC	198		Clear L
	LDX	0	PT2	
THRU	DEC	30	6	Test constant
SMSK	DEC	30	63	Mask for N-sector
B00001	HEX	FFB8	CD87	Boston
B00002	HEX	FAD5	A7DA	Atlanta
B00003	HEX	FC12	CA08	Chicago
B00004	HEX	FFDA	55AC	Dallas
B00005	HEX	BA7C	2CC8	San F(rancisco)
B00006	HEX	FFFF	FE7D	End
NT5	DEC	24	5	Repeat Count of 5
ALLF	HEX	FFFF	FFFF	Mask
PT2	INP	0	BACK	Input course/distance or wind
BACK	EXC	398		U to L, L to U
	TMI	ADD	DEE	If high order character greater than 7, go to ADD
DEE	EXC	398	BIN	U to L, L to U
ADD	ADU	2AT3		
	TMI		DUN	If high order character is /, go to DUN
	SBU	2AT3	DEE	
DUN	EXC	398		U to L, L to U
	XCXE	0		Turn on Branch Control
	XTBC	W00001	0	

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
BIN	MPT	98		Binarize
	CLU	RECRC0		
	SRL	104		
	ADU	RECRC0	BACK	
W00001	CLU	CRSE	GO	Store course
W00002	CLU	EDIST	DOWN	Store distance
W00003	CLU	DIR	GO	Store wind direction
W00004	CLU	VEL	OUT	Store wind velocity
GO	XLDX	1	BACK	Step X
DOWN	XLDX	1	PT2	Step X
OUT	RAU	CRSE	STOB	Compute angle A
STOB	SBU	DIR		
	TMI	PT3		Go to PT3 if wind direction greater than bearing
	SBU	18E		
	TMI	CMPL	USE	
CMPL	CLU	RECRC0		
	SBU	RECRC0	USE	
18E	DEC	31	180	Constant
2AT3	DEC	3	2	Test constant
PT3	ADU	18E		
	TMI	CMPL	USE	
USE	STU	RECRC7		Store angle A
	RAL	ALLF		
	CME	ZERO		
	TBC	TAIL		
	CME	18E		
	TBC	HEAD		
	CMG	9T		
	TBC	SINE		Go to SINE if A 90 degrees or less
	RAU	18E		
	SBU	RECRC7	SINE	
SINE	SRL	114		
	EXC	498		U to X
	XRAU	S00001		Get sine of angle
	STU	RECRC4		Sin A @ 1
	MPY	VEL		w sin A @ 32
	DIV	CCC		w sin A / a = sin W @ 1
	STU	V00001		Store sin W for a = 300
	SRL	1		
	STU	V00002		Store sin W for a = 600
	ADU	V00001		
	STU	V00003		Store sin W for a = 200
	SRL	1		
	STU	V00004		Store sin W for a = 400
	MPY	JUG	PT4	

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
JUG	DEC	0	.8	Constant
CCC	DEC	31	300	Constant
9T	DEC	31	90	Constant
PT4	STU	V00005	AGIN	Store sin W for a = 500
AGIN	RAL	ALLF		Mask
	LDX	0		
	LDC	NT63		
	CMG	S00001		Search sine table for sin W
	TBC	ANGL		If found, go to ANGL
	LDX	100		
	LDC	NT26		
	XCMG	S00001		Search upper sine table for sin W
	TBC	ANGL	ERRO2	If not found, go to ERRO2
ANGL	CLU	RECRC6		Store sin W
	EXC	898		X to U
	SBU	NS1		
	EXT	SMSK		Get found sector
	SRL	113		
	SAU	DB1QIK	DB2QIK	Set XLDX address to found sector
DB2QIK	XLDX	0		Set X to relative location in table
	XRAU	S00001	PT5	Get comparing table entry
ZERO	HLT	0	0	Constant
NS1	DEC	30	1	Constant
NT26	DEC	24	26	Repeat Count of 26
NT63	DEC	24	63	Repeat Count of 63
PT5	XSBU	S00000		Subtract next lower entry
	CLU	RECRC0		Save entry interval
	RAU	RECRC6		Get sin W
	XSBU	S00000		Subtract next lower table entry
	DIV	RECRC0		Compute interpolation factor
	TBC		NOVR	If no overflow, go to NOVR
	EXC	898		X to U
	SRL	14		
	ADU	RECRC7		Compute A plus W
	RAL	ALLF		
	CMG	9T		
	TBC	SHL		If 90 degrees or less, go to SHL
	CLU	RECRC5		
	RAU	18E		
	SBU	RECRC5	SHL	Compute angle G
SHL	SRL	114		
	EXC	498		U to X
	XRAU	S00001	STOW	Get sin G
NOVR	CLU	RECRC0		
	EXC	898		X to U
	SRL	14		

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	ADU	RECRC7		Compute A plus W (high)
	RAL	ALLF		
	CMG	9T		
	TBC	SFT		If 90 degrees or less, go to SFT
	STU	RECRC5		
	RAU	1AT0		
	SBU	RECRC0		Complement interpolation factor
	CLU	RECRC0		
	RAU	18E		
	SBU	RECRC5	SFT	Compute angle G (low)
SFT	SRL	114		
	EXC	498		
	XRAU	S00002	PT6	Get sin G (high)
1AT0	HEX	8000	0000	
PT6	XSBU	S00001		Compute G (low) - G (high) interval
	MPY	RECRC0		Interpolate
	XADU	S00001	STOW	Compute sin G
STOW	CLU	RECRC0		Save
	LDC	ENTRA		
	RAU	P00001		Get airspeed
	ADU	VEL		Add wind velocity
	STU	RECRC1		Save
	SBU	VEL		
	STU	RECRC2		Save airspeed minus wind velocity
	ADU	VEL		
	MPY	RECRC0		Compute a sin G
	DIV	RECRC4		a sin G / sin A
	RAL	ALLF		
	CMG	RECRC1		
	TBC		TAIL	If computed ground speed greater than airspeed plus wind, go to TAIL
	CMG	RECRC2		
	TBC	HEAD	PT7	If computed ground speed less than airspeed minus wind, go to HEAD
P00001	DEC	31	300	Airspeed table
P00002	DEC	31	600	
P00003	DEC	31	200	
P00004	DEC	31	400	
P00005	DEC	31	500	
ENTRA	DEC	24	4	Repeat Count parameter
PT7	LDC	ENTRA	SAGS	
SAGS	STU	G00001	DECR	Store ground speed
TAIL	LDC	ENTRA		
	RAU	P00001		Get airspeed
	ADU	VEL	PT7	Compute ground speed
HEAD	LDC	ENTRA		
	RAU	P00001		Get airspeed
	SBU	VEL	PT7	Compute ground speed
DECR	RAU	ENTRA		Get Repeat Count parameter

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	SBU	NT1		Reduce by 1
	TMI	RSET		If negative, go to RSET
	STU	ENTRA		Store
	RAU	RECRC7		Get angle A
	CME	ZERO		
	TBC	TAIL		If A = 0, go to TAIL
	CME	18E		
	TBC	HEAD		If a = 180, go to HEAD
	LDC	ENTRA		
	RAU	V00001	AGIN	Get next sin W and return to AGIN
RSET	ADU	NT5		Reset Repeat Count parameter to 4
	STU	ENTRA		
	RAU	EXSW		Reverse exit/return switch
	RAL	TERM		
	STL	EXSW	PT8	
NT1	DEC	24	1	
PT8	STU	TERM		
	RAU	EGSSW	TERM	
TERM	SAU	SAGS	ENEXT	
GOON	RAU	CRSE		Compute course from destination to base
	SBU	18E		
	TMI		STOB	
	ADU	36T	STOB	
EGSSW	HLT	EGSNB	0	
EXSW	SAU	SAGS	GOON	
36T	DEC	31	360	

SUBROUTINE E

We come now to the subroutine which will select, on the basis of the ground speeds provided by Subroutine T, the aircraft which will provide the fastest service to the destination. It is entered from Subroutine T each time the ground speeds for one base are computed.

The method of selection is this. The base reference number is used as a search key to bring from the Aircraft Status Table each entry for this base. When a table word is brought into the accumulator, the A/C type is extracted and used as a key to find the range from the 5-word range table and the ground speed from the table prepared by **Subroutine T.**

The time enroute is now computed and checked against the range for this A/C type. If the range is insufficient, this A/C is eliminated from the selection process and we return to select another A/C from the Aircraft Status Table. If an A/C has sufficient range, its time enroute is

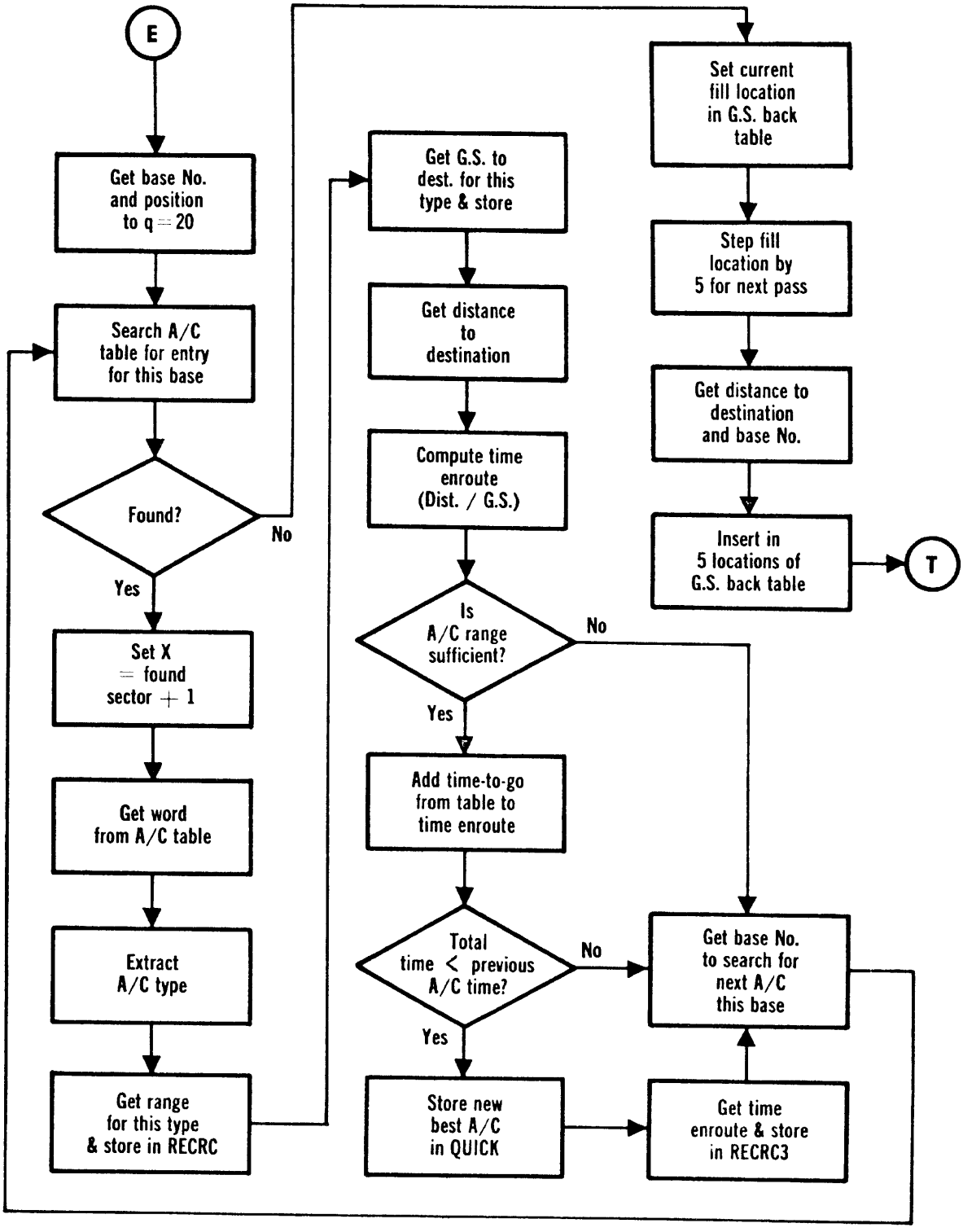


FIGURE 11. Flow Chart - Example Subroutine E

to its time-to-go value from the Aircraft Status Table, and this sum is checked against the last computed best time. If smaller, this new best time replaces the previous one and we return for the next A/C from the Aircraft Status Table.

When all A/C assigned to this base have been processed, the distance and base number are stored in five locations of the Ground Speed Back Table. These five locations contain the ground speeds back to base for five A/C types previously computed by Subroutine T. The new information is stored under control of a mask so as to leave these ground speeds undisturbed.

When finished, control is returned to Subroutine T to process the next base.

Aircraft Status Table Search

The coding of Subroutine E contains a search of the Aircraft Status Table for all entries associated with a particular base. At location ENEXT, the base reference number is brought from ESAVB. It is then positioned to agree with the base field in the table and a corresponding mask is placed in the LOWER.

The INDEX Register is set initially to zero, and at PASS a Repeat Count of 24 is set up. The XCME will now be executed in Repeat Mode, and for the first pass will begin its search with the initial location of the table. If no matching base is found, we branch to NOMO. If a match is found, the found sector plus 1 is exchanged into the UPPER and compared with a termination constant stored at NS25. For the first search, the constant will always be equal to or greater than the found sector plus 1, which is now shifted to a "q" of 17 and used to set the INDEX value. Since this value is one greater than the relative table location with which we are concerned, we may bring the correct word from the table with an XRAU whose Data-address is one less than the initial location. In ROAR coding, we can express this initial location minus 1 for Region A as A00000.

The A/C type from the table word is used as a key to select the proper range and ground speed. The coding sequence from location PE2 to PE3 computes time enroute, checks for sufficient range and determines whether the A/C presently being processed is a better choice than the previous A/C. At LONG, the base number is again placed in the UPPER. The mask is brought back into the LOWER and we return to PASS for the next table search.

Note that each time we return to begin a new search, the INDEX Value will be equal to the relative table location at which the search is to begin. Since we will always use a Repeat Count of 24 for this search, it is possible that we may find a match which is beyond the end of the table. If this occurs, the found sector plus 1 will be greater than 25 and will be detected by the comparison with the termination constant, NS25. In this event, we exit from the loop to location NOMO.

At NOMO, we get from EGSSW the location at which the ground speeds back to this base are stored. This same address is then used as the Data-address of GSNB. The EGSSW Data-address is incremented by 5 to establish the storage block for the next execution of Subroutine T. The distance and base number are inserted into the table of ground speeds back to base with a Masked Store (MST) command and control is returned to Subroutine T.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	E		Compute new shortest time to destination
ENEXT	RAU	ESAVB		Get base reference
	SRL	112		Position
	EXT	BASS		
	STU	BASR		Save base reference
	RAL	HEXB		Mask
	LDX	0	PASS	
PASS	LDC	NT24		
	XCME	A00001		Search for available A/C
	TBC		NOMO	Go to NOMO if finished
	EXC	898		X to U
	EXT	SMSK		Get found sector plus 1
	RAL	SMSK		
	CMG	NS25		Test for end of table
	TBC		NOMO	If finished, go to NOMO
	SRL	113		
	EXC	498		U to X
	XRAU	A00000		Get table entry
	EXT	HEXT		Extract A/C type
	SRL	9		Position
	STU	RECRC1	PE2	Store Repeat Count
HEXT	HEX	7	0000	Mask for A/C type (0 to 4)
NS25	DEC	30	25	Constant
SMSK	DEC	30	63	Mask for N-sector
NT24	DEC	24	24	Repeat Count of 24
HEXB	HEX	8000	E000	Mask for base (1 to 5) and availability bit
BASS	HEX	0	E000	
PE2	LDC	RECRC1		
	RAU	R00001		Get A/C range in minutes
	CLU	RECRC0		
	LDC	RECRC1		
	RAU	G00001		Get ground speed to destination
	CLU	RECRC1		Store @ 31
	RAL	EDIST		Get distance to destination @ 63
	SRL	107		Distance @ 56
	DIV	RECRC1		Compute time enroute @ 25
	MPY	6T		Convert to minutes @ 31

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	RAL	TMSK		Mask
	CMG	RECRC0		Test for sufficient range
	TBC		LONG	If no, go to LONG
	XADU	A00000		Add time-to-go
	CMG	QUICK		Compare with previous time to destination
	TBC		LONG	Go to LONG if previous time smaller
	STU	QUICK		Store new smaller entry
	XSBU	A00000	PE3	Subtract time-to-go
TMSK	HEX	0	1FFF	Mask for time-to-go
6T	DEC	6	60	Constant
R00001	DEC	31	600	10 hours
R00002	DEC	31	300	5 hours
R00003	DEC	31	480	8 hours
R00004	DEC	31	240	4 hours
R00005	DEC	31	300	5 hours
PE3	CLU	RECRC3	LONG	Save time enroute
LONG	RAU	BASR		Get base reference
	RAL	HEXB	PASS	Get mask and return for next pass
NOMO	RAU	EGSSW		
	SAU	GSNB		Set next storage address in GSNB
	ADU	DS5		Increment by 5
	CLU	EGSSW		Save
	RAL	EDIST		Get distance
	SRL	116		
	RAU	HEXB		Get mask
	MML	BASR		Merge distance and base
	RAU	HEXX		
	LDC	ENTRA	GSNB	
GSNB	MST	98	TBELO	Store in EGSNB table
HEXX	HEX	1FFF	E000	Mask
DS5	DEC	17	5	Constant
QUICK	HEX	7FFF	FFFF	A/C status word storage

SUBROUTINE Q

When the end of an input message is detected by Subroutine T, control is transferred to Subroutine Q. At this point, the best available A/C will have been selected, and its logged time, A/C number, A/C type, current base and time-to-go to destination placed in location QUICK. If no A/C of sufficient range is available, QUICK will contain all 1's except for a 0 sign bit.

Subroutine Q will test the contents of location QUICK and, if service is not available will exit to Subroutine S. Otherwise, it will proceed to

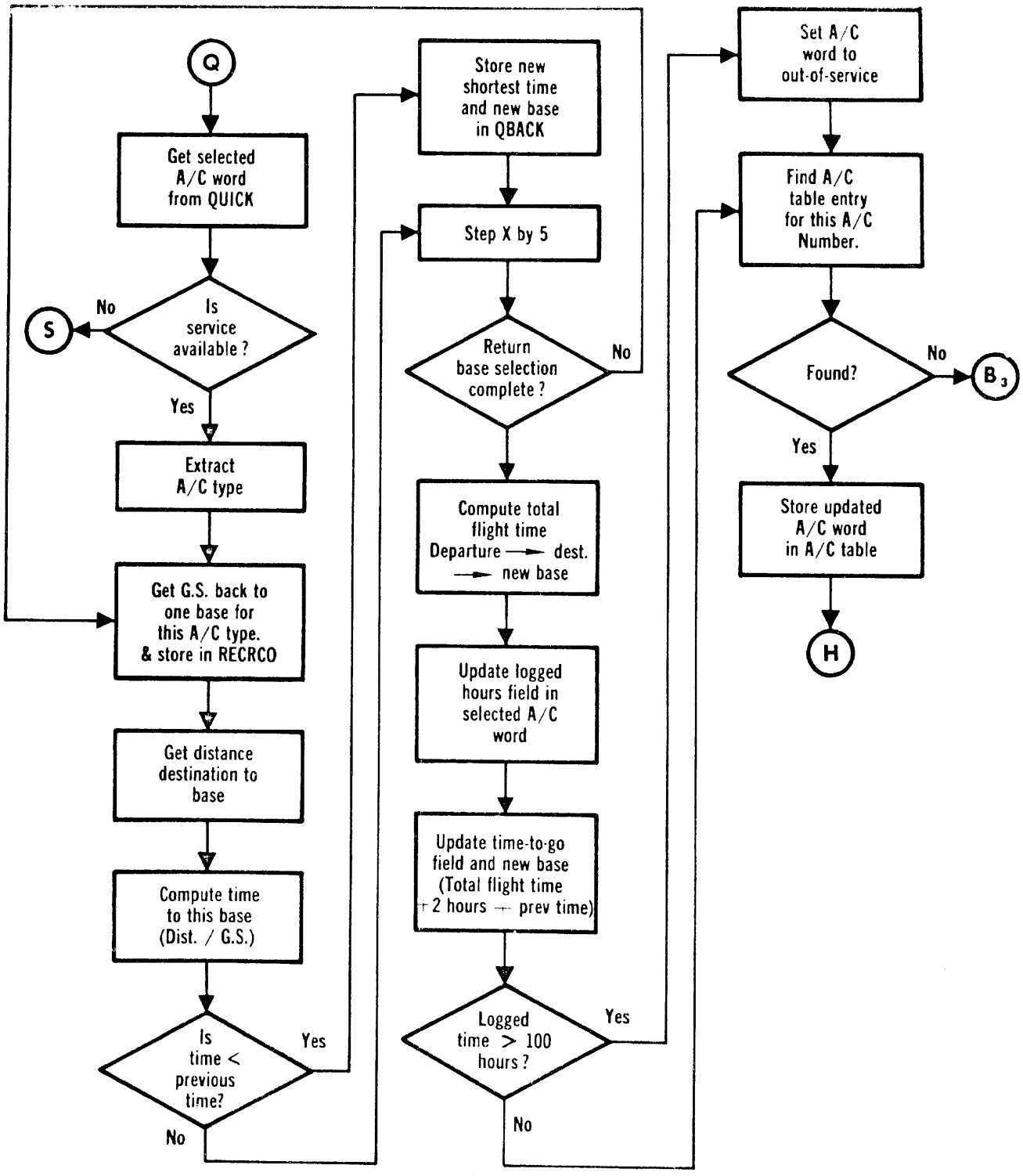


FIGURE 12. Flow Chart - Example Subroutine Q

select a new base and update the appropriate entry in the Aircraft Status Table.

The A/C type from QUICK is used as a key to get the ground speed from destination to each base. For each prospective new base, a time enroute is computed and compared against the previous shortest time. If the new time is shorter, it replaces the old shorter time, the Index Value is stepped by 5, and a test is made to determine whether all bases have been considered. If not, another pass through the loop is executed.

When the best new base is found, the logged hours field in QUICK is updated by adding the total flight time from base to destination to new base. The time-to-go field is updated by adding to its current value, the total flight time plus two hours ground time, and the base reference number is changed to that of the newly assigned base. If the new logged hours value is greater than 100, the availability indicator is set to out-of-service, which will apply at the conclusion of this assignment.

The correct Aircraft Status Table location for this A/C number is now determined by a search of the table. If not found, we exit to an error halt in Subroutine B. When the location is found, the updated entry in QUICK is stored in the status table and we exit to Subroutine H.

Table Structure for Ground Speeds to New Base

When Subroutine Q is executed, it will choose a new base assignment on the basis of ground speeds stored in the 25 word table beginning at location EGSNB. This table consists of five 5-word blocks, each block containing the ground speeds and distance back to one base for 5 A/C types.

The INDEX Register is set initially with the A/C type from location QUICK and may be any value 0 thru 4. The XRAU instruction at location AGAN will bring into the UPPER, the ground speed and distance to one base for the selected A/C type. Subsequent references to the table will be in 5-word increments to select the same type for each of the five bases. When the INDEX Register has been stepped to or beyond 25 at location OLD, we will drop through the TMI instruction at OLD plus 3, instead of returning to AGAN for another pass.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	Q		Select new base. Update A/C Status Table
NOMOR	RAU	QUICK		Get selected A/C word
	RAL	ALLF		Get mask

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	CME	HEXQ		Test for service available
	TBC	NOTE1		If no, go to NOTE1
	EXT	HEXT		Extract A/C type
	SRL	2		
	EXC	498	AGAN	U to X
AGAN	XRAU	EGSNB		Get ground speed and distance
	EXT	TMSK		Get ground speed
	CLU	RECRC0		Save ground speed @ 31
	XRAL	EGSNB		Get ground speed and distance
	SRL	16		Discard all but distance
	SRL	107		Distance @ 56
	DIV	RECRC0		Compute time in hours @ 25
	MPY	6T		Time minutes @ 31
	STU	RECRC0		Save
	RAU	TMSK		Get mask in U
	RAL	ALLF		Get mask in L
	EXT	QBACK	PQ2	Get previous time to new base
QBACK	HEX	7FFF	FFFF	A/C word storage for new base
6T	DEC	6	60	Constant
TMSK	HEX	0	1FFF	Mask for ground speed
HEXT	HEX	7	0000	Mask for A/C type (0 to 4)
HEXQ	HEX	7FFF	FFFF	A/C word mask
ALLF	HEX	FFFF	FFFF	Mask
PQ2	CMG	RECRC0		Compare with new time to new base
	TBC	OLD		Go to OLD if old value smaller
	RAL	RECRC0		Get new shorter time
	RAU	HEXB		Get mask
	XMML	EGSNB		Merge base with time
	CLL	QBACK	OLD	Store new smaller time and base
OLD	XLDX	5		Step X by 5
	EXC	898		X to U
	SBU	DS25		Test for finish
	TMI	AGAN		If not, return to AGAN
	RAU	TMSK		Mask
	EXT	QBACK		Time to new base @ 31
	ADU	RECRC3		Add time enroute @ 31
	DVU	NT60		Convert to hours @ 7
	ADU	1AT8		Round
	EXT	LMSK		Extract whole hours
	ADU	QUICK		Add A/C Status word for total logged hours
	STU	RECRC3		
	EXT	HEXN		Extract all but base
	ADU	QBACK	PQ3	Compute total flight time to new base
HEXN	HEX	FFFF	1FFF	Mask for everything but base
LMSK	HEX	FF00	0000	Mask for hours
1AT8	DEC	8	1	Constant

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
NT60	DEC	24	60	Constant
DS25	DEC	17	25	Constant
HEXB	HEX	8000	E000	Mask for base (1 to 5) and availability bit
PQ3	ADU	2HRS		Add ground time
	STU	RECRC6		Save
	ADU	2T8		Test for logged time 100 hours or more
	TMI		LESS	If not, go to LESS
	STU	RECRC6	LESS	Store and set to -out of service-
LESS	RAL	AMSK		Mask
	LDC	NT24		
	CME	A00001		Find table entry
	TBC		ERRO3	If A/C not found, go to ERRO3
	EXC	898		X to U
	EXT	SMSK		
	SRL	113		
	EXC	498		U to X
	RAU	RECRC6		Get A/C word
	XCLU	A00000	HEADS	Update A/C Status table and go to HEADS
SMSK	DEC	30	63	Mask for N-sector
NT24	DEC	24	24	Constant
AMSK	HEX	F8	0000	Mask for A/C number
2T8	DEC	7	28	Constant
2HRS	DEC	31	120	Constant

SUBROUTINE H

Having completed the selection of the aircraft to be dispatched and the new base, we are ready to output, via the typewriter, an Aircraft Assignment Message. Subroutine H performs the function of printing the message headings.

A carriage return and two line feeds are output and the INDEX Register is set to 0. A word from the output table is placed in the UPPER and a mask is placed in the LOWER. The leftmost character from the UPPER is printed and the contents of UPPER and LOWER are shifted left six places. We now test for completion of one word and, if not, we return to print the next character. When one table word is printed, we step the INDEX Register by 1 and check for completion of the headings. If not, we return to get the next word from the table. When all headings are printed, we exit to Subroutine P.

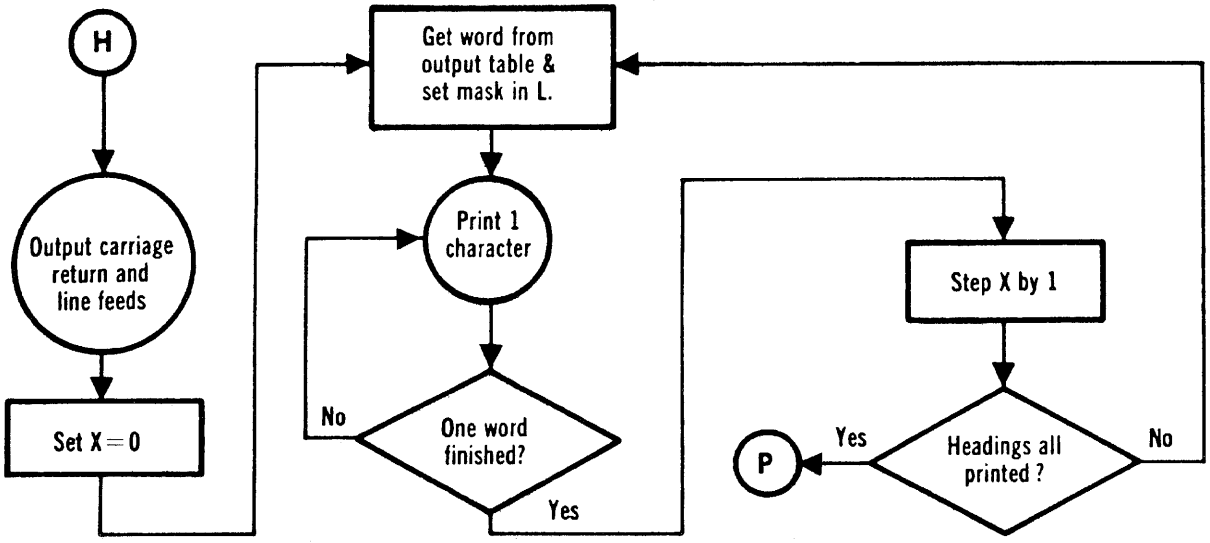


FIGURE 13. Flow Chart-Example Subroutine H

The Print Loop

The headings are printed by means of a fairly simple double loop. The output print image is contained in a table stored in Region H. Each word contains five 6-bit characters, with the last two bits of each word blank. The necessary tabs and spaces are included as characters.

At location GRAB, a table word is brought into the UPPER and a mask is placed in the LOWER. At ALPH, a character is printed. The Data-track 64 designates that the six-bit character to be printed is in the UPPER. The artificial Data-sector 99 causes ROAR to assign a sector which will honor the print interlock. When the next character is positioned by the shift, the low order six bits of the LOWER are filled with zeroes. The CME instruction will not turn on Branch Control until all five characters have been printed and shifted out of the UPPER. At this point, the mask in the LOWER will contain ones only in the two leftmost positions; the corresponding positions of the UPPER will contain zeroes.

When this inner loop is completed, the INDEX Register is stepped and tested for the value 10. If less than 10, we return to GRAB for another pass of the outside loop. When the Index Value reaches 10, the CXE will turn on Branch Control and we will exit to Subroutine P.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	H		Print message headings
HEADS	PRD	199		Carriage return
	PRD	799		Line feed
	PRD	799		
	LDX	0	GRAB	
GRAB	XRAU	H00001		Get heading word
	RAL	ALLF	ALPH	Mask
ALPH	PRU	6499		Print one character
	SRL	106		Position to next character
	CME	ZERO		Test finish of one word
	TBC		ALPH	If no, go to ALPH
	XLDX	1	PH2	Step X
ZERO	HLT	0	0	Constant
ALLF	HEX	FFFF	FFFF	Mask
H00001	HEX	75EA	5AAC	DEPAR
H00002	HEX	B6EA	5E08	TURE Tab.
H00003	HEX	75EB	2D88	DESTI
H00004	HEX	9DAB	62A0	NATIO
H00005	HEX	9C27	AD68	N Tab. ETA
H00006	HEX	0A77	B0F4	Tab. NEW Sp.
H00007	HEX	6DAB	1E08	BASE Tab.
H00008	HEX	7AD6	8268	ETA Tab A
H00009	HEX	F9CF	67A0	/C sp. NO
H00010	HEX	F011	C700	. C.R. L.F. L.F.
PH2	CXE	10		Test end of heading print
	TBC	PRINT	GRAB	If no, go to GRAB

SUBROUTINE P

The Aircraft Assignment Message, exclusive of headings, is printed by Subroutine P. The destination print sequence is also used in printing a service-not-available message.

The departure base is printed first in the following manner. The base number is used to set the INDEX Register to provide a reference to the desired entry in the Base Print Image Table. The base print image is then brought into the UPPER and LOWER Accumulators, and the INDEX Register is set to 0. Characters are now printed from the combined accumulators, using the INDEX Register as a character counter. When all characters have been printed, we check to determine whether the base is SAN FRANCISCO. If so, it is necessary to print one more word from the table. When the entire departure base is printed, we encounter a 2-position program switch set to follow path number 1.

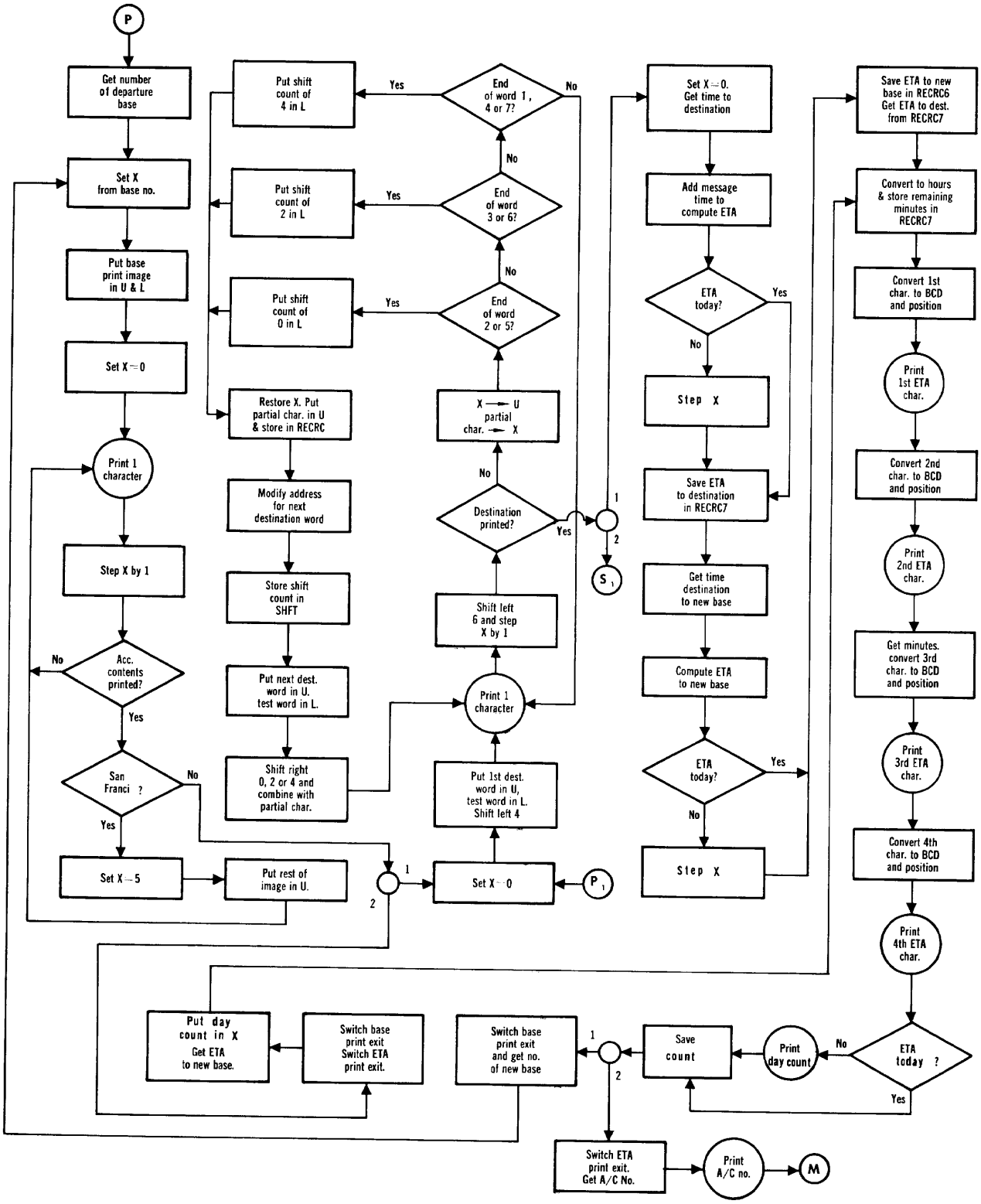


FIGURE 14. Flow Chart - Example Subroutine P

The INDEX Register is set to 0 and the first word of the destination print image is placed in the UPPER. The image table capacity is eight words and may contain a destination of up to 42 6-bit characters. These characters are stored contiguously, with vacant character positions at the high order end of the table. With this storage format, some characters will be split between two table words.

A test word is placed in the LOWER which will be used to detect the printing of the last full character from the UPPER. The first character is positioned to bits 0-5 of the UPPER and a PRU instruction is executed. If the character position contains all zeroes, it represents a tape feed code and causes no typewriter output. The next character is positioned and the INDEX Register is incremented by 1. A test is made for completion of 42 characters and, if not, the INDEX Register and UPPER Accumulator contents are exchanged. Now a sequence of tests determines whether the last full character of a word has been printed. If not, UPPER Accumulator and INDEX Register contents are again exchanged and the next character is printed.

When the last full character of a word is printed, any remaining partial character is saved and a shift is set up to be used in combining it with the next table word. When the next word is brought in from the image table, it is shifted right by the number of bits in the partial character and combined with it. The next character is printed and the above procedure continues until the INDEX Value reaches 42. At this point, the destination printing is complete and we encounter a 2-position program switch set to follow path number 1.

The next sequence will compute the ETA's to destination and new base as follows. The INDEX Register is reset to 0, and the time to destination is brought into the UPPER where it is added to the time of the message. If this ETA is not the same day as the message, a day count is set in the INDEX Register. The ETA to destination is saved in RECRC7 and the ETA to new base is computed in a similar manner and stored in RECRC6.

Now the ETA to destination is brought back from RECRC7, converted to hours and minutes, and the minutes stored in RECRC7. The hours are converted from binary to BCD and printed. The minutes are brought from RECRC7, converted to BCD and printed. If this ETA is not the same day as the message, a "+" is printed followed by the day count.

Another program switch is encountered and again we follow path number 1. Now the program switch which serves as an exit from the base print sequence is switched to follow path number 2. The number of the new base is brought into the UPPER and we return to the base print sequence to print the new base.

This time when we reach the exit, we branch to a sequence in which the base print exit is reset to its initial value, and the ETA print exit switch is set to follow path number 2. The ETA to new base is brought into the UPPER and we return to the ETA print sequence.

Upon completion of the ETA to new base, we branch to a point where the ETA exit switch is reset, and the A/C number is brought into the UPPER. The A/C number is converted to BCD form and printed and we exit to Subroutine M.

There is a secondary entry point to Subroutine P which is entered from Subroutine S in connection with a "service not available" output message. As a part of this message, we need to print the destination. The destination print exit switch will have been set by Subroutine S to follow path number 2. When the destination is printed, we now exit back to Subroutine S, where the exit switch will be reset to its initial value.

Program Switches

The technique of programmed exit switches is used at three points in Subroutine P. These switches occur at location PP2, PEXIT and SAFE. The procedure is simply a matter of replacing the original contents of the switch with an alternate instruction word prior to execution of the required instruction sequence. Upon completion of this sequence, the switch is reset to its original value.

The Destination Print Sequence

At location PDEST, the Index Value is set to 0. The INDEX Register will be used as a counter to control the exit after 42 executions of the PRU instruction at LETR. The print image is stored in Region D, with the first word in D00008 and the last word in D00001. The storage format is such that the first four bits of the first word will always be blank. Bits 28-31 of words 1,4 and 7 contain partial characters. Bits 30-31 of words 3 and 6 contain partial characters. The low order end of words 2 and 5 coincide with the end of a character.

As each print image word is brought into the UPPER, a flag containing 1's in bit positions 5 - 17 is brought into the LOWER. When a character is printed at location LETR, the combined UPPER and LOWER contents are shifted left by six bits. When the last full character of any word has been printed, this shift will move the flag to a position in the UPPER such that bits 5 - 17 will contain a track and sector value, 12763, 3163 or 763. The flag is exchanged into the INDEX Register at PEXIT plus 1, and the series of CXE instructions will test for one of these three values.

If one of these tests is positive, the associated TBC will branch to a point where a Shift Count of 0, 2 or 4 is brought into the LOWER. The INDEX Register contents are exchanged back into the UPPER and the leftmost four bits are extracted and stored in RECRC5 to save any partial character they may contain.

The Data-address of DB1GET is modified to the next image word reference and the Shift Count in the LOWER is stored in location SHFT. When the new image word and the flag are brought into the accumulators, they are shifted right by 0, 2 or 4 bits and combined with the partial character from RECRC5 at location JOIN. The Next-address of JOIN branches back to LETR to print the next character.

ETA Conversion for Printing

To print on ETA, it is necessary to convert from minutes in binary form to hours and minutes in BCD form. At location ENTR, the ETA value will have been placed at a "q" of 63 in the LOWER Accumulator. The UPPER is cleared and the ETA shifted to a "q" of 62. Dividing by a value 60 at a "q" of 31 will compute the whole hours at a "q" of 31, leaving the remainder, in minutes, in the LOWER.

The minutes are saved in RECRC7 and the hours divided by a value 10 at a "q" of 28, which converts the first character to BCD and moves it to a "q" of 3. A rounding constant is added and the first character is printed. This character is now eliminated by an EXT instruction, the remaining character is multiplied by ten, and printed as the second hour digit. Now the minutes are brought back from RECRC7, converted and printed in the same manner. The day count in the INDEX Register is checked and, if set, a "+" is printed followed by the day count.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	P		Output A/C assignment message
PRINT	RAU	HEXB		Mask
	EXT	QUICK	POS	Get departure base
POS	SRL	101		Position
	EXC	498		U to X
	XRAU	M00000		Get print image 1.
	XRAL	M00005		Get print image 2.
	LDX	0	CHAR	
CHAR	PRU	6499		Print one character
	XLDX	1		
	CXE	10		Test for end of image 2.
	TBC	PP2		If yes, go to PP2
	SRL	106	CHAR	Position next character
M00001	HEX	6E8B	2DA2	BOSTO(N)
M00002	HEX	64D9	5A9E	ATLAN(T)
M00003	HEX	7218	9C6A	CHICA(G)
M00004	HEX	75A9	656A	DALLA(S)
M00005	HEX	B1A9	FD7E	SAN F(R)

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
M00006	HEX	7F7D	F420	(N) sp. sp. sp. Tab.
M00007	HEX	D6BD	F420	(T)A sp. sp. Tab.
M00008	HEX	0A3D	F420	(G)O sp. sp. Tab.
M00009	HEX	CF7D	F420	(S) sp. sp. sp. Tab.
M00010	HEX	B6A7	7220	(R) ANCI
M00011	HEX	B1CA	3D08	SCO sp. Tab.
HEXB	HEX	8000	E000	Mask for base (1 to 5) and availability bit.
PP2	TMI	HO	PDEST	If finished, go to PDEST
HO	LDX	5		Reset X
	RAU	M00011	CHAR	Get print image 3 for San Francisco
PDEST	LDX	0		
	RAU	D00008		Get word 1
	RAL	FLAG		Test word
	SRL	104	LETR	Position first character
LETR	PRU	6499		Print one character
	SRL	106		Position to next character
	XLDX	1		Step X
	CXE	42	PEXIT	Test for end of message
PEXIT	TBC	PFINI		If yes, go to PFINI
	EXC	1298		X to U, U to X
	CXE	12763		Test for end of word 2 or 5
	TBC	SHO		If yes, go to SHO
	CXE	3163		Test for end of word 3 or 6
	TBC	SH2		If yes, go to SH2
	CXE	763		Test for end of word 1, 4 or 7
	TBC	SH4		If yes, go to SH4
	EXC	1298	LETR	X to U, U to X
SH4	RAL	SFT4	PP3	
SH2	RAL	SFT2	PP3	
FLAG	HLT	12763	0	Termination flag
SFT2	SRL	2	EX2	Shift constant
SFT4	SRL	4	EX4	Shift constant
SHO	RAL	SFTO	PP3	Get Shift Count
PP3	EXC	1298		X to U, U to X
	EXT	LFT4		
	STU	RECRC5		Save partial character
	RAU	DB1GET		Modify address for next word
	SBU	DS1		
	CLU	DB2GET		
	CLL	SHFT	DB1GET	Set Shift Count
DB1GET	RAU	D00008		Get next word
	RAL	FLAG	SHFT	Test word
SHFT	SRL	4	JOIN	Position
EX4	EXT	X4	JOIN	
EX2	EXT	X2	JOIN	
JOIN	ADU	RECRC5	LETR	Add partial character

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
PFINI	PRD	299		Print Tab.
	LDX	0		
	RAU	RECRC3		Get time to destination @ 31
	EXT	TMSK		
	ADU	MTIME	PP4	Compute ETA in minutes
TMSK	HEX	0	1FFF	Mask for time-to-go
X2	HEX	3FFF	FFFF	
X4	HEX	0FFF	FFFF	
DS1	HLT	1	0	Constant
LFT4	HEX	F000	0000	Mask
SFT0	SRL	0	JOIN	Shift Constant
PP4	EXC	198		U to L
	SBU	DAY		Subtract 24 hours
	TMI	TDAY		If ETA today, go to TDAY
	XLDX	1	PP4	Step count
TDAY	STL	RECRC7		Save ETA to destination
	RAU	TMSK		Mask
	EXT	QBACK		Get time - destination to new base
	ADU	RECRC7		ETA to destination plus time to new base
	ADU	1HR		Compute ETA to new base
	EXC	198		U to L
	SBU	DAY		Subtract 24 hours
	TMI	SDAY		If same day as ETA to destination, go to SDAY
	XLDX	100		Step counter
	EXC	198	SDAY	U to L
SDAY	CLL	RECRC6		Save ETA to new base
	RAL	RECRC7	ENTR	Get ETA to destination @ 63
ENTR	CLU	JUNK		
	SRL	101		ETA @ 62
	DIV	1HR		Convert to hours @ 31
	CLL	RECRC7		Save minutes
	DIV	TEN		Position hours
	ADU	ONE	PP5	
TEN	DEC	28	10	Constant
1HR	DEC	31	60	Constant
DAY	DEC	31	1440	24 hours in minutes
PP5	PRU	1699		Print first character
	EXT	X4		
	MPT	98		Convert character
	PRU	1699		Print second character
	RAU	RECRC7		Get minutes
	DVU	TEN		Position minutes
	ADU	ONE		
	PRU	1699		Print third character
	EXT	X4		

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	MPT	98		Convert character
	PRU	1699		Print fourth character
	CXE	0		Are both ETA's today
	TBC	TOD		If yes, go to TOD
	CXE	100		Is ETA to destination today
	TBC		TOM	If no, go to TOM
	XLDX	12701	TOD	Reduce index to 1
TOM	PRD	5899		Print +
	EXC	898		X to U
	SRL	111		
	TMI		SHIF	If both ETA's same day, go to SHIF
	XLDX	12701	SHIF	Modify counter
SHIF	SRL	103		Position day count
	PRU	1699	TOD	Print day count
TOD	PRD	299	PP6	Print Tab.
ONE	DEC	31	1	Constant
PP6	EXC	898	SAFE	X to U
SAFE	CLU	RECRC7	SWCH	Save day count
SWCH	RAU	ETAN		Switch exit to print new base
	CLU	PP2		
	RAU	HEXB		
	EXT	QBACK	POS	Get new base reference
NETA	RAU	THAW		Switch exit initial setting
	CLU	PP2		
	RAU	ACNO		
	CLU	SAFE		
	RAU	RECRC7		Get day count
	EXC	498		U to X
	RAL	RECRC6	ENTR	Get ETA to new base
AC	RAU	SWIT		Switch ETA print exit to SWCH
	CLU	SAFE		
	RAU	AMSK		Mask
	EXT	QUICK		Get A/C number
	DIV	X		Position
	ADU	ONE	PP7	
X	DEC	9	10	Constant
AMSK	HEX	F8	0000	
SWIT	CLU	RECRC7	SWCH	
ACNO	CLU	RECRC7	AC	
THAW	TMI	HO	PDEST	
ETAN	TMI	HO	NETA	
PP7	PRU	1699		Print first character
	EXT	X4		
	MPT	98		Convert character
	PRU	1699		Print second character
	PRD	199	MAINT	Carriage return

SUBROUTINE M

When an Assignment Request Message or an Availability Message has been processed, Subroutine M is entered and a count is made of the A/C in maintenance. If more than 12 A/C are out of service, a note is printed via the typewriter.

The count, which has been tallied in the LOWER, is exchanged to the UPPER where it is converted to BCD and printed as a 2-digit number. A print loop is now entered which will take the remainder of the message from a print image table. When the message is complete, control is transferred to Subroutine B where the program is initialized in preparation for a new message.

In the event that fewer than 13 A/C are out of service, the print sequence is bypassed and we proceed directly to Subroutine B.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	M		Print note if less than 13 A/C in service
MAINT	CLL	JUNK		Clear L for count
	LDX	0	GET	
GET	XRAU	A00001		Get A/C word
	TMI		STEP	If in service, go to STEP
	ADL	ONE	STEP	Add 1 to count
STEP	XLDX	1		Step X
	CXE	25		Test for finish
	TBC		GET	If no, go to GET
	EXC	298		L to U
	SBU	3TN		Test if more than 12 A/C in maintenance
	TMI	SENSE		If no, go to SENSE
	ADU	3TN		
	DVU	TEN		Position
	ADU	ONE		
	PRD	199		Carriage return
	PRD	799		Line feed
	PRD	799		Line feed
	PRU	1699		Print first character
	EXT	RT28		
	MPT	98	PM2	Convert character
RT28	HEX	0FFF	FFFF	Mask
TEN	DEC	28	10	Constant
3TN	DEC	31	13	Constant
ONE	DEC	31	1	Constant
PM2	PRU	1699		Print second character

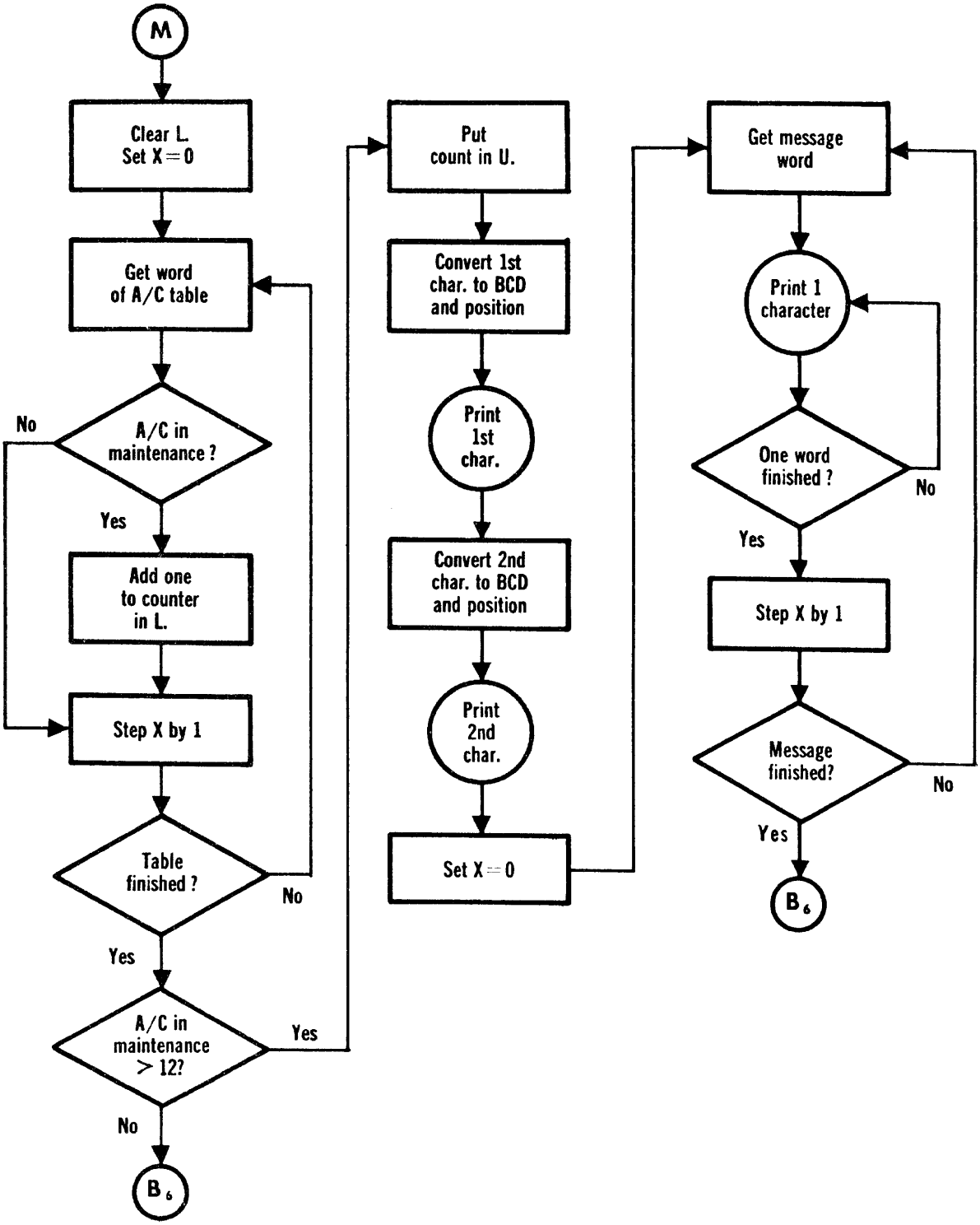


FIGURE 15. Flow Chart - Example Subroutine M

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	LDX	0	GETT	
GETT	XRAU	N00001		Get message word
	RAL	ALLF	LTTR	Mask
LTTR	PRU	6499		Print one character
	SRL	106		Position
	CME	ZERO		Test finish of one word
	TBC		LTTR	If no, go to LTTR
	XLDX	1		Step X
	CXE	5		Test for end of message
	TBC	SENSE	GETT	If no, go to GETT
ZERO	HLT	0	0	Constant
ALLF	HEX	FFFF	FFFFF	Mask
N00001	HEX	F5A8	AB70	sp. AIRC
N00002	HEX	ADA7	EDF4	RAFT sp.
N00003	HEX	8A7F	6668	IN sp. MA
N00004	HEX	8A7B	5E9C	INTEN
N00005	HEX	6A77	1E04	ANCE C.R.

SUBROUTINE S

If it is determined in Subroutine Q, that no available A/C has sufficient range to reach a destination, Subroutine S is entered. The first part of a "service not available" message is printed from a print image table.

When this is complete, the exit switch for the destination print sequence in Subroutine P is set to as to return control to Subroutine S, and control is transferred to print the destination. When Subroutine S is re-entered, the switch is reset to its initial value and we exit to Subroutine B for initialization.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	S		Print note if service not available
NOTE1	PRD	199		Carriage return
	PRD	799		Line feed
	PRD	799		
	LDX	0	GIT	
GIT	XRAU	C00001		Get message word
	RAL	ALLF	LTRR	Mask
LTRR	PRU	6499		Print one character
	SRL	106		Position

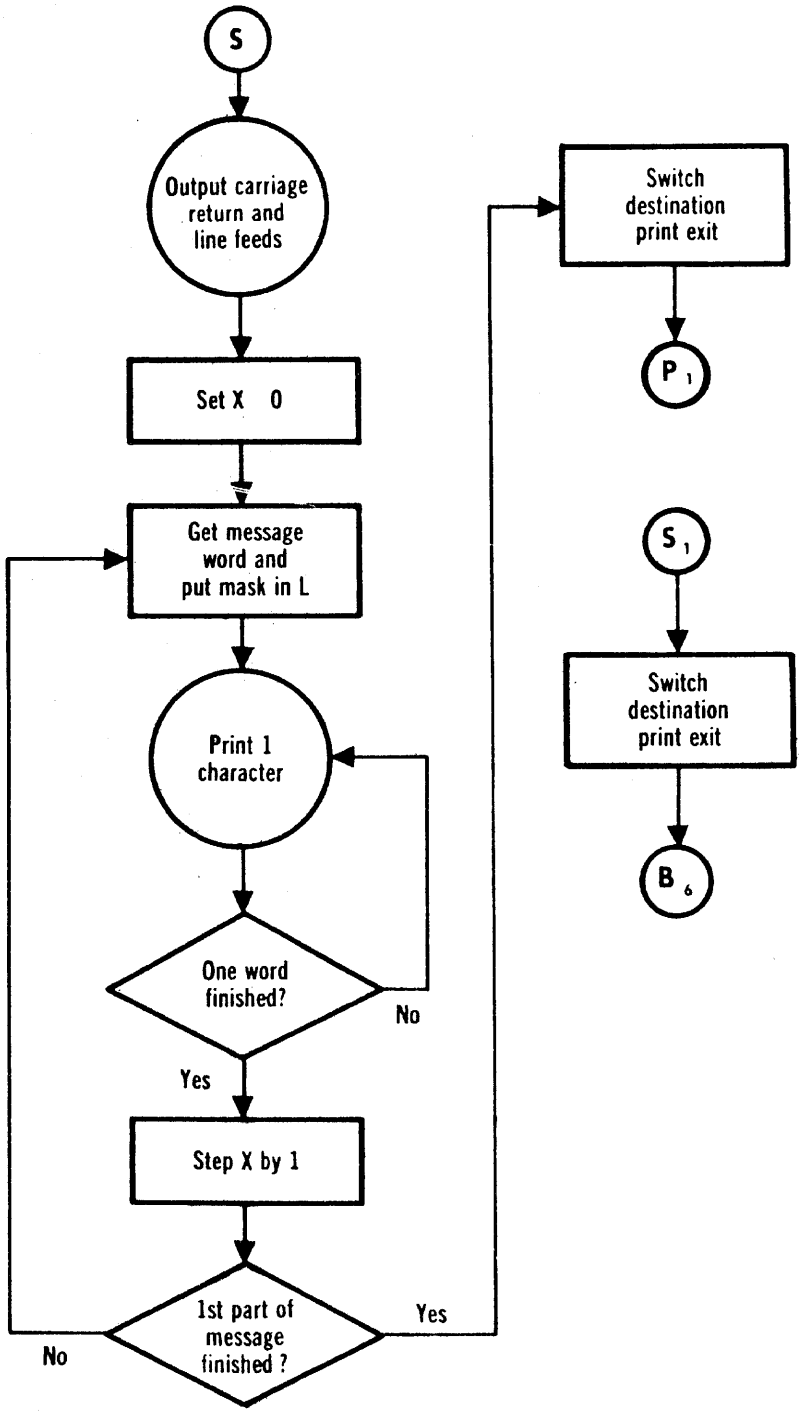


FIGURE 16. Flow Chart - Example Subroutine S

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	CME	ZERO		Test finish of one word
	TBC		LTRR	If no, go to LTRR
	XLDX	1		Step X
	CXE	5		Test for end of message
	TBC		GIT	If no, go to GIT
	RAU	EXIT		Switch destination print exit
	SAU	PEXIT	PDEST	Go to destination print
TAGS	RAU	FISH	PS2	Switch destination print exit
EXIT	HLT	TAGS	0	
ZERO	HLT	0	0	Constant
ALLF	HEX	FFFF	FFFF	Mask
C00001	HEX	B1EA	EF88	SERVI
C00002	HEX	71EF	67A0	CE sp. NO
C00003	HEX	B7D6	AF68	T sp. AVA
C00004	HEX	8A56	9B94	ILABL
C00005	HEX	7BDB	68F4	E sp. TO sp.
PS2	SAU	PEXIT		
	PRD	199	SENSE	Carriage return
FISH	HLT	PFINI	0	

SUBROUTINE V

When Subroutine R detects an Availability Message input, Subroutine V is entered. The message word, which contains the A/C number is brought into the UPPER and the A/C number is converted from its 6-bit input form to binary. This number is used as a key to search the Aircraft Status Table. If no entry is found for this A/C number, we exit to an error halt in Subroutine B.

When the table location for this A/C is found, it is used to set the INDEX Value. The message word is brought back into the UPPER, and its last character is tested for a "U". If the test is positive, this A/C is set to "unavailable" in the Aircraft Status Table.

If the last character of the input message is not a "U", it is tested for an "A". If not an "A", we exit to an error halt in Subroutine B. If it is an "A", the Aircraft Status Table entry is modified to make the A/C available for assignment.

At the completion of Subroutine V, control is transferred to Subroutine M to count the number of A/C in maintenance.

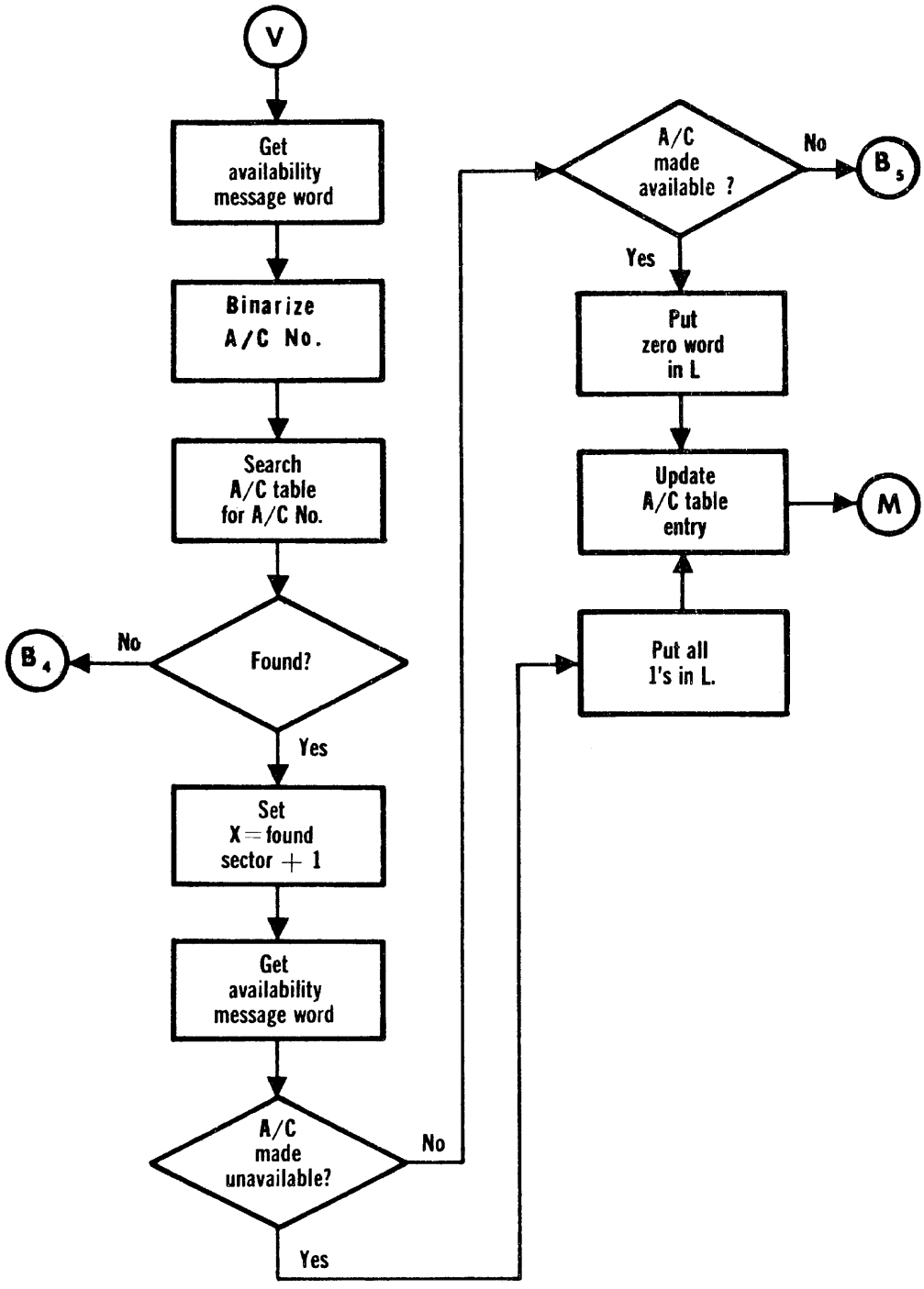


FIGURE 17. Flow Chart - Example Subroutine V

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	V		Updata A/C table for availability message
AVMES	RAU	D00001		Availability message
	EXT	4BIT		Get characters in BCD
	SRL	18		Position for binarization
	MPT	98		Binarize
	CLU	RECRC3		
	SRL	106		
	ADU	RECRC3		
	SRL	119		Shift A/C number to table position
	RAL	AMSK		Get mask
	LDC	NT24		Repeat Count
	CME	A00001		Find table entry
	TBC		ERRO4	If A/C not found, go to ERRO4
	EXC	898		X to U
	EXT	SMSK		Mask
	SRL	113		Position
	EXC	498		U to X
	RAU	D00001		Availability message
	RAL	3F		Mask
	CME	UCON	PV2	Test A/C un-availability
3F	HEX	0	003F	Mask
SMSK	DEC	30	63	Mask for N-sector
NT24	DEC	24	24	Constant
AMSK	HEX	F8	0000	Mask
4BIT	HEX	3C	F000	Mask
PV2	TBC	YOU		If unavailable, go to YOU
	CME	ACON		Test A/C availability
	TBC		ERRO5	If no, go to ERRO5
	RAL	ZERO	GEB	Get zero value
GEB	RAU	BITS	PLUG	Mask
PLUG	XMST	A00000	MAINT	Store in table entry
YOU	RAL	ALLF	GEB	Get word of all 1's
ALLF	HEX	FFFF	FFFF	Constant
BITS	HEX	FF00	1FFF	Mask
ZERO	HLT	0	0	Constant
ACON	HEX	0	001A	Constant
UCON	HEX	0	002E	Constant

SUBROUTINE B

The first subroutine of the program to be executed is the last one to be coded. Subroutine B performs the initialization of various tables and instructions which are modified during the operation of the program. It also contains the five error halts previously discussed.

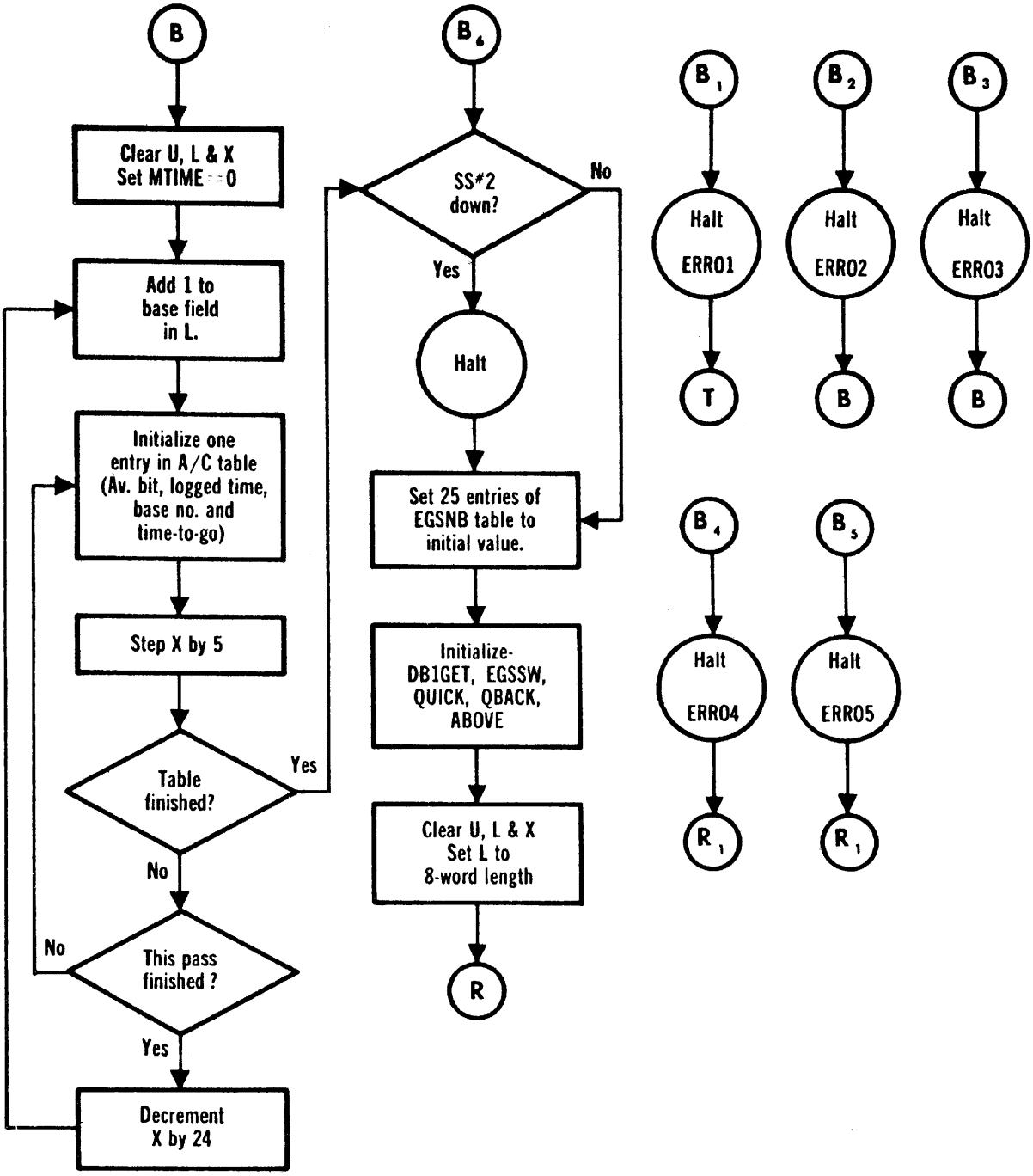


FIGURE 18. Flow Chart - Example Subroutine B

When Subroutine B is entered at START, the UPPER and LOWER Accumulators and the INDEX Register are cleared and the message time in location MTIME is set to 0. The base number 1 is set up in the LOWER and along with a zeroed availability bit, logged time and time-to-go is stored into every fifth location of the Aircraft Status Table under control of a mask so as to leave the A/C numbers and types intact. The base number is increased to 2 and we again store into every fifth location, this time beginning with the second word of the table. This process is repeated until the entire table has been initialized.

Sense Switch 2 is now tested and, if down, a programmed halt is encountered. The table of ground speeds from destination to base is now initialized by storing a value in every location which will indicate to Subroutine Q any base for which ground speeds have not been computed.

The parameters, QUICK, QBACK and EGSSW are now initialized as are the Data-address fields of the instruction words DB1GET and ABOVE. The accumulators and the INDEX Register are again cleared and control is transferred to Subroutine R.

For subsequent executions of the program, Subroutine B is entered at the point indicated as B_6 in the flow chart.

If Subroutine B is entered at B_1 it indicates that an illegal base has been input.

Continuing from the halt will return control to Subroutine T where a correct base may be input.

If Subroutine B is entered at B_2, an unsuccessful Sine Table search is indicated. If entered at B_3, an unsuccessful search for an A/C number in the Aircraft Status Table is indicated. Both cases are considered incorrigible errors and continuing from the halt will cause a return to location START.

If Subroutine B is entered at B_4 or B_5, it indicates an illegal Aircraft Availability Message format. Continuing from the halt will transfer control to Subroutine R where a corrected message may be input.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
	TAG	B		Initialization and error stops
START	CLU	JUNK	OVER	
ERRO1	HLT	1	TBELO	Departure base not found in table
ERRO2	HLT	2	START	Sine Table search unsuccessful
ERRO3	HLT	3	START	Dispatched A/C No. not found in Status Table

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
ERRO4	HLT	4	MREAD	Availability A/C No. not found in Status Table
ERRO5	HLT	5	MREAD	Last character of Availability Message (U or A) may be in error
OVER	EXC	3798		U to X, U to L, L = 1
	CLU	MTIME	MORE	Set message time to zero
MORE	ADL	BAS1	GEM	Step base number
	RAU	MSK1		Mask
	XMST	A00001		Reset table word
	XLDX	5		Step X
	CXE	29		Test for finish
	TBC	SENSE		If yes, go to SENSE
	EXC	898		X to U
	ADU	OVRF		Overflow if finished one pass
	TBC		GEM	If not, go to GEM
	XLDX	12740	MORE	Decrement X by 24
SENSE	SNS	200		Test Sense Switch 2
	TBC	PB2	PROC	If not down, go to PROC
OVRF	HEX	7FF9	C000	Overflow constant
MSK1	HEX	FF00	FFFF	Mask for availability bit, logged hours, base number and time-to-go
BAS1	HEX	0000	2000	1 @ 20
PB2	HLT	0	PROC	
PROC	RAU	DONE		1 @ 31
	LDC	NT24	STEW	
STEW	STU	EGSNB		Initialize EGSNB Table
	RAU	RSET		
	SAU	DB1GET		Initialize DB1GET
	RAU	RSET		
	SAU	EGSSW		Initialize EGSSW
	RAU	HEXQ		
	STU	QUICK		Initialize QUICK
	CLU	QBACK		Initialize QBACK
	SAU	ABOVE		Initialize ABOVE
	EXC	598		Clear X and L
	EXC	1699	INPUT	L = 8
HEXQ	HEX	7FFF	FFFF	Initializing constant
REST	HLT	EGSNB	0	Initializing constant
RSET	HLT	D00008	0	Initializing constant
NT24	DEC	24	24	Repeat Count of 24
DONE	HEX	1FFF	0001	Initializing constant

AIRCRAFT STATUS TABLE AND SINE TABLE FORMATS

The Aircraft Status Table is stored in Region A and the table values are entered using the HEX pseudo-instruction. Initially the aircraft number, type, and assigned base are entered. During the operation of the program the aircraft number and type will retain their original table storage, but the base number may change with the processing of an Aircraft Assignment Message.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS					
A00001	HEX	0009	2000	A/C	1	Type	1	Base	1
A00002	HEX	0011	4000		2		1		2
A00003	HEX	0019	6000		3		1		3
A00004	HEX	0021	8000		4		1		4
A00005	HEX	0029	A000		5		1		5
A00006	HEX	0034	2000		6		4		1
A00007	HEX	003C	4000		7		4		2
A00008	HEX	0044	6000		8		4		3
A00009	HEX	004C	8000		9		4		4
A00010	HEX	0054	A000		10		4		5
A00011	HEX	005B	2000		11		3		1
A00012	HEX	0063	4000		12		3		2
A00013	HEX	006B	6000		13		3		3
A00014	HEX	0073	8000		14		3		4
A00015	HEX	007B	A000		15		3		5
A00016	HEX	0080	2000		16		0		1
A00017	HEX	0088	4000		17		0		2
A00018	HEX	0090	6000		18		0		3
A00019	HEX	0098	8000		19		0		4
A00020	HEX	00A0	A000		20		0		5
A00021	HEX	00AA	2000		21		2		1
A00022	HEX	00B2	4000		22		2		2
A00023	HEX	00BA	6000		23		2		3
A00024	HEX	00C2	8000		24		2		4
A00025	HEX	00CA	A000		25		2		5

The table of sines is 91 words in length and is stored in Region S. Values are entered using the DEC pseudo-instruction.

LOCATION	ORDER	DATA ADDRESS	NEXT ADDRESS	COMMENTS
S00001	HLT	0	0	Table of sines beginning with sin 0
S00002	DEC	1	.0175	
S00003	DEC	1	.0349	
.	.	.	.	
.	.	.	.	
.	.	.	.	
S00090	DEC	1	.9998	
S00091	DEC	1	1.000	End of table - sin 90

PROGRAM TESTING

The program which we are discussing was tested first for the correct operation of each of the subroutines. This was accomplished by punching and assembling each subroutine as an independent program. The listing was scanned for references to Regions and 5-character symbolic addresses outside the subroutine being tested. Simulated values for these tables, parameters and instruction words were included in the symbolic tape before assembling.

The assembled subroutine was loaded into memory along with the utility programs Change and Transfer and Sequential Memory Print. The subroutine was executed and Sequential Memory Print used to print out the contents of all tables and locations which were altered by the subroutine. This data was then checked for correctness against precomputed values.

In several cases where the subroutine failed to function properly, it was possible to locate the error by executing a portion of the subroutine in One-Operation Mode while verifying the register contents from the oscilloscope. In other cases it was necessary to re-analyze coding sequences in the light of the printed results.

The Change and Transfer Program provides the ability to set or alter any word in memory via the typewriter. This function was used to make on-the-spot corrections which could be verified before repunching a corrected tape. It was also used to bring selected words from memory into the LOWER Accumulator for examination.

Output Subroutines

Output subroutines were tested, for the most part, by verifying their output against expected results. It was sometimes necessary to simulate

initial conditions prior to testing the subroutines. The simulations were designed to test the operation over the full range of possible conditions.

Final Testing

When all subroutines had been successfully tested individually, they were combined into a single punched symbolic tape. This tape was then assembled by ROAR and loaded into the computer.

A program input tape was prepared containing a variety of Aircraft Assignment Request Messages intermixed with Availability Messages which gradually removed all aircraft from service. Each output was checked for correct assignments and ETA's. Removal of aircraft from service permitted checking the "Aircraft in Maintenance" output message and the "Service Not Available" message.

The final testing phase brought to light a few errors in the intercommunication linkage between subroutines which were readily corrected to produce a finished program tape.

MANUAL CONTROLS AND OPERATING PROCEDURES

6

The RPC-4000 Electronic Computing System is composed of the RPC-4010 Computer and the RPC-4500 Tape/Typewriter System.

Other input/output devices may be added in various combinations to a maximum of 60 devices.

The list of components:

RPC-4000	Electronic Computing System
RPC-4010	Computer
RPC-4500	Tape/Typewriter System
RPC-4480	Typewriter
RPC-4430	Reader/Punch
RPC-4410	Photo-Electric Reader
RPC-4440	High Speed Punch
RPC-4431	Auxiliary Reader/Punch
RPC-4600	Auxiliary Tape/Typewriter System
RPC-4450	Line Printer

A magnetic tape unit is planned and other devices will be announced as they become available. This discussion will concern itself with the basic RPC-4000 system, the RPC-4010 Computer, and the RPC-4500 Tape/Typewriter System.

The RPC-4010 computer console contains the controls for the central computer. The RPC-4500 Tape/Typewriter System consisting of the RPC-4480 Typewriter and the RPC-4430 Reader/Punch provide means for presenting information to the central computer via the typewriter keyboard or the reading unit on the RPC-4430 Reader/Punch. The central computer can output information via the typewriter or the punching unit on the RPC-4430. The master controls for all input/output devices are located on the right panel of the RPC-4430.

A list of all the controls on both the RPC-4010 and the RPC-4430 may be found on pages 130 through 138 .

The control panels on the units are referred to as:

RPC-4010 Computer Console Control Panel - abbreviated - (CC)
RPC-4430 Reader/Punch Right Control Panel - abbreviated - (RPR)
RPC-4430 Reader/Punch Left Control Panel - abbreviated - (RPL)

THE RPC-4010

The RPC-4010 console has buttons that control the central computer. The use of the SENSE SWITCHES is discussed on pages 29 and 132. The POWER ON and POWER OFF buttons do not require discussion.

MODES OF COMPUTER OPERATION

If the ONE OPERATION button is raised, the computer is in NORMAL mode.

If the ONE OPERATION button is depressed, the computer is in ONE OPERATION Mode.

Normal Mode

If conditions are satisfied for the operation of a given program and the computer is in Normal Mode, a start signal will initiate the execution of program steps until a programmed halt is encountered or the ONE OPERATION button is depressed. During computer operation in the Normal Mode, the COMPUTE indicator will generally be lighted during the execution of program steps. INP, PRD, or PRU commands may cause the STOP indicator to light and the COMPUTE indicator to go out. However, so long as the computer is under program control, manual intervention is not required unless it pertains to input of information. When ONE OPERATION is depressed, the computer stops in the next phase 3 that is encountered.

If an instruction is being executed in the Repeat Mode, the computer will perform the number of executions specified by the Repeat Count (an extended phase 4 will occur) before stopping in the following phase 3 (See page 25 .)

During Normal Mode the computer will stop when a HLT command is executed or when ONE OPERATION is depressed.

The HLT command is similar to INP in that the effect of the command occurs in phase 3. Ordinarily the instruction word in the COMMAND Register has not been executed. In the case of HLT, obviously it has been

executed if the computer stops. When a programmed HLT is encountered the computer stops in phase 3 of the HLT cycle with the HLT instruction word in the COMMAND Register.

One Operation Mode

When the ONE OPERATION button is depressed the computer halts at the beginning of the next phase 3 that is encountered, initiating One Operation Mode. This computer mode is used for bootstrapping and while debugging a program.

In One Operation Mode a start signal will cause the instruction in the COMMAND Register to be executed and the next instruction in the program to be placed in the COMMAND Register.

Certain conditions can exist which will modify the operation of an instruction.

a. If the sign of the UPPER Accumulator is negative and the COMMAND Register contains a TMI instruction word the next instruction to be executed will be designated by the data-address of the COMMAND Register.

b. If Branch Control is on and the COMMAND Register contains a TBC instruction word the next instruction to be executed is designated by the data-address. If the BRANCH CONTROL button is depressed, resetting (or turning OFF) Branch Control, the next instruction will be designated by the next-address of the COMMAND Register as usual.

c. A programmed HLT executed when ONE OPERATION is depressed is redundant.

When the RPC-4010 is in One Operation Mode the SET INPUT and EXECUTE LOWER ACCUMULATOR buttons may be activated.

SET INPUT

When SET INPUT is depressed the instruction in the COMMAND Register is overridden by an "effective" input command. The COMMAND Register is unchanged except for the data-track field which is set to zero. The "effective" input command is treated in the same way as any other instruction is treated in One Operation Mode. The computer halts at the beginning of phase 3 of the "effective" input command which will not be executed until START COMPUTE (RPR) (CC) is depressed.

If EXECUTE LOWER ACCUMULATOR is depressed when the "effective" input command is executed, the next start signal received as the result of the detection of a stop code by the selected input device or the depression of START COMPUTE (RPR) (CC), will cause the contents of the LOWER Accumulator to be transferred to the COMMAND Register, by-passing the usual phase 1 and 2 and the computer to halt with the contents of the LOWER Accumulator. The instruction word now in the COMMAND Register will not be executed until the another start signal is received.

EXECUTE LOWER ACCUMULATOR should not be depressed unless input is being accomplished by means of SET INPUT.

THE RELATIONSHIP OF ONE OPERATION AND REPEAT MODES

In One Operation Mode it is possible that the Repeat Count will be loaded and Repeat Mode initiated. There are several possibilities that may occur.

The effect of Repeat Mode when the ONE OPERATION button is raised is explained in Section 3. Only a brief review is necessary. The repeat Count is loaded into bits 18 thru 24 of the INDEX Register by the LDC instruction word. The next phase four that is encountered is executed and then repeated the number of times specified by the Repeat Count (Repeat Count ÷ word times for execution = number of repetitions).

When ONE OPERATION is depressed and an LDC instruction word is executed, the instruction word designated by the next-address is transferred to the COMMAND register and the computer stops at the next phase 3 that is encountered. Thus the Repeat Count will be intact in the INDEX Register. If no other buttons are activated the instruction now in the COMMAND Register will be executed in the Repeat Mode if it could be so executed in Normal Mode. All of the restrictions that apply in Normal Mode will apply in One Operation Mode.

The possibility does exist that manual operations can intervene.

a. One Operation Mode-Repeat Count not loaded. Obviously no problem.

b. One Operation Mode- Repeat Count loaded.

If the last instruction to be executed was an LDC and the instruction now in the COMMAND Register can be executed in the Repeat Mode, the next start signal will cause the instruction to be executed just as if the computer were in Normal Mode. Since only phase 4 of an instruction is repeated all of the repetitions will occur before the computer stops in phase three of the following instruction.

c. One Operation Mode-Repeat Count loaded-EXECUTE LOWER ACCUMULATOR depressed.

If the last instruction to be executed was an LDC and EXECUTE LOWER ACCUMULATOR was depressed when the computer entered the next phase 1, the contents of the LOWER Accumulator was transferred to the COMMAND Register and the computer halted in phase 3 of the instruction word now in the COMMAND Register. If this instruction would normally be executed in the Repeat Mode the next start signal will initiate its repeated execution as in b. (above)

d. One Operation Mode-Repeat Count loaded-EXECUTE LOWER ACCUMULATOR depressed-SET INPUT depressed.

If SET INPUT is depressed after the Repeat Count is loaded, the Repeat Mode is terminated and the Repeat Count remains intact in the INDEX Register. The instruction following SET INPUT is executed only once regardless of actual input or whether the instruction is taken from memory or the LOWER Accumulator.

Certain commands cannot be repeated. Any command which does not have a phase 4 cannot be repeated for obvious reasons (active TBC and TMI, INP, HLT). These commands would alter the Repeat Count during a repeated execution: LDC, EXC (UPPER into INDEX) and would cause a phase 4 that would never stop, hence they will not be repeated if so attempted by the program.

PROTECTED TRACKS

The RPC-4010 memory may be protected to prevent destruction of recorded information in 8 blocks of 16 tracks each. Two blocks (tracks 0-15, 16-31) may be protected by means of toggle switches located behind the sliding panel directly below the oscilloscope on the RPC-4010 console. These switches disconnect the write heads to prevent recording. (Figure 19.)

Inside the RPC-4010 cabinet are 8 toggle switches that will protect any of the 8 blocks of 16 tracks. These switches are mounted on circuit cards located inside the front panel of the RPC-4010 (and behind the metal door covering the circuit card racks). Eight toggle switches are mounted two each on 4 cards. The upper switch on the left-most card is in series with the 0-15 switch on the computer console, the lower switch is in series with the 16-31 switch on the console.

The eight switches inside the cabinet control the write-heads (reading

upper, lower, left to right) on tracks:

0-15	64-79
16-31	80-95
32-47	96-111
48-63	112-127

If the two internal switches (0-15, 16-31) corresponding to the external switches are off, the external switches have no effect. If one is off, both are off. If both are on, recording is possible. The switches are in series.

ADDITIONAL COMMENTS ON MODES

A summary of the various modes may be useful:

One Operation and Normal Mode can only be selected manually (by the ONE OPERATION button).

Copy Mode can be controlled manually or by programming.

The Eight-Word Mode of the LOWER Accumulator is controlled by programming, but depressing the SET INPUT button in One Operation Mode will set the LOWER to one-word length, regardless of its programmed mode.

The Repeat Mode can be controlled by programming only.

Single Character Mode means that after each four-bit or six-bit character is read as determined by the INP command, a start signal will be sent to the computer. Single Character Mode cannot be controlled by programming and with the exception of SET INPUT, the four-bit or six-bit input cannot be controlled manually.

The selection of four-bit or six-bit input is under control of the INP (x) command, except for the effect of SET INPUT.

THE RPC-4000 INPUT/OUTPUT

The basic RPC-4000 system uses the RPC-4500 Tape/Typewriter System consisting of the RPC-4480 Typewriter and the RPC-4430 Reader/Punch. The RPC-4430 Right Control Panel has the master INPUT/OUTPUT controls for any device in the system. The control buttons and indicators are listed and described on pages 134 through 138.

THE RPC-4430

The RPC-4430 has two panels. The right panel (with the exception of SYSTEM POWER) is concerned exclusively with on-line operations. The left panel is actually in two sections divided by the caption OFF LINE. The top portion deals primarily with on-line operation, but must be considered to affect off-line functions to a certain extent. The principle function of the lower portion is to control off-line operation, but it can have a decided effect on on-line operation.

Selection On and Off Line

Any unit selected off-line takes precedence over on-line selection. Selection of the typewriter, the reader or the punch off-line will deselect that unit if it is selected on-line. If the unit is selected off-line and any attempt (by programming or manually) is made to select that unit on-line, the SELECTION MONITOR indicator lights to indicate an error. The light under the on-line selection button will come on and with the exception of the SELECTION MONITOR indicator, the unit will appear to be selected on-line. THE COMPUTER WILL BE UNABLE TO USE UNITS SELECTED BOTH AND OFF-LINE. They may be used off-line in this condition.

One and only one input device may be selected at any one time. If an input device is selected on-line manually or by a program step and another input device is subsequently selected, the input that was selected last will be the one that is used by the RPC-4010. The lights under both buttons will be on and both will appear to be selected but only the last will have access to the RPC-4010. If two devices are so selected the SELECTION MONITOR indicator (RPL) will be lighted.

The RPC-4000 Tape

The tape-reader moves the tape under a set of "read brushes" which close an electrical circuit when a tape hole (punch) passes beneath them. The tape has seven channels. Any or all of these channels may have a punch in one horizontal row. A combination of punches in channels 1 through 6 is called a character. Channel 7 is called the parity bit.

Parity Checking

The PARITY MONITOR circuit (RPR) is used to check the accuracy of the reader and of the punch which produced the tape that is being read.

That is, channel 7 of the tape is reserved for a parity bit. The RPC-4000 system has even parity. If the number of punches in a character is odd, a parity bit will be punched so that the number of bits in all seven channels is always even. The parity checking circuitry is active only during on-line operation of the tape-reader.

If the PARITY MONITOR INHIBIT button is depressed, parity checking is by-passed. If the button is raised, the reader will check the parity of the tape being read.

If the PARITY MONITOR INHIBIT button is raised and a character with an odd number of bits is read, the light under the PARITY MONITOR RESET button comes on and input stops. At this point the STOP READ button (RPR) must be depressed. The character with bad parity is the character that was read before the one that is now in the CHARACTER INDICATOR lights. The character with bad parity has been transferred to the RPC-4010.

If the parity error is such that the character so input is invalid, the accumulators must be cleared and all information input as a result of the INP being executed must be read into the computer again.

Recovery from a parity stop is normally as follows:

depress STOP READ (RPR),
depress PARITY MONITOR RESET (RPR),
clear the accumulators,
read all of the input again,
re-position the tape, if necessary,
depress START READ

When START READ is depressed, the character under the read-head (and also in the CHARACTER INDICATOR LIGHTS is input to the RPC-4010 computer just as if the parity stop had not occurred.

Checksums and Master Tapes

The advantage of checking the reliability of information input into a computer is widely recognized. One method was discussed above. A second method is checksumming.

Various schemes are used to produce a checksum. The criteria for choosing a scheme depends on the characteristics of the various components in the computing system, the assemblers that produces the machine language program tape and the input routines. Using these criteria, the following scheme was adopted for the RPC-4000. The RPC-4000 Optimizer and Assembly Routine, ROAR, which will produce most RPC-4000 program master tapes, outputs a checksummed hexadecimal master tape of the following format:

The bootstrap followed by a blank space,
several tape records (structured as below) and
a transfer record.

For present purposes, a tape word is defined as the characters between stop codes. The structure of each tape record of the hexadecimal program tape (except the final record) is 100 tape word of 12 hexadecimal characters, each followed by a stop code. The final word (101) is a checksum of not more than 16 hexadecimal characters.

In the tape word, the first 4 characters are an absolute address and the remaining eight characters are the information to be entered at the specified address. Leading zero's in the first 4 characters are omitted.

In the checksum (word 101), 16 characters may be present (leading zeroes are supressed). The first 8 characters are the sum of all the locations (first 4 characters) in the preceding words in the record plus the total number of overflows detected while summing the data words (last 8 characters) in the record. The final 8 characters in the cheskcum are the sum (without regard to overflow) of all the data words in the record.

The final tape record is of variable length and is the same as other records except that the checksum of this final record will always have bit 0 equal 1 and will always have 16 characters. Only the final record will ever have a checksum with negative sign bit.

The last record on the tape is a transfer record to branch to the beginning of the program being loaded. If SENSE SWITCH 32 is depressed the transfer will occur immediately and the program will begin operation.

If SENSE SWITCH 32 is raised a halt will occur before a transfer to the starting location of the program.

COPY MODE

Copy Mode means that whatever is being input to the RPC-4010 will be copied on all output devices that are selected by the RPC-4010. Two buttons on the RPC-4430 Right Control Panel control Copy Mode. They are the INPUT DUPLICATION SELECT and RESET buttons. Copy Mode may also be controlled by the appropriate PRD instruction. Although Copy Mode is an on-line activity and may be under program control it is a function of the RPC-4430. If the RPC-4010 selects more than one output device and prints or punches on these devices simultaneously, this is multiple output and is a function of the RPC-4010. Such multiple output should not be confused with Copy Mode.

The RPC-4480 Typewriter

The RPC-4480 typewriter may be used as input/output device on-line or off-line. It could also be used as a conventional typewriter.

Before the typewriter can be used (either on-line or off-line), it must be selected. On-line selection may be manual or by the appropriate PRD instruction. Off-line selection overrides on-line selection as stated on page 123.

If the typewriter is used for output, the only action required by the operator is setting the tab stops and margins.

On-line Operation

If the RPC-4480 is being used for on-line input, it must be selected (manually or by programming), the computer must have executed an INP command or SET INPUT (CC) and START COMPUTE (RPR) (CC) must have been depressed.

When the above conditions are met, the RPC emblem on the upper left of the face of the type bar rack cover will be lighted. This light indicates that the RPC-4010 is ready to accept input and information may be typed.

If the SINGLE CHARACTER MODE button (RPR) is raised, the computer will accept codes 16 through 62, that is all the legal alpha-numeric characters. (See page 185). When (and only when) a stop code is detected, at start signal is sent to the computer. If SINGLE CHARACTER MODE (RPR) is depressed, all character codes are entered into the computer and a START signal is sent after each character is read.

A stop code (*) is a special character representing a configuration of bits that the RPC-4430 normally recognizes (on and off-line) as a signal to stop input and take some other action. Single Character Mode (on or off-line) allows the treatment of stop codes to be identical to that of any other character.

When an input device is selected on-line and SINGLE CHARACTER MODE (RPR) is raised and the RPC-4430 recognizes a stop code originating from the device, the RPC-4430 terminates input and sends a start signal to the RPC-4010.

Off-line Operation

When an input device is selected off-line and the CONDITIONAL STOP button (RPL) is raised, a stop code that originates with the input device

and is detected by the RPC-4430 will cause the RPC-4430 to act as if STOP READ (RPL) had been depressed. If CONDITIONAL STOP (RPL) is depressed, stop codes will be treated in the same way as any other character.

Operation of the RPC-4480

The typewriter keyboard is similar to any good electric typewriter. The operation is very similar and it only seems necessary to comment on certain differences.

The numeric keys include a "1".

The "*" is the stop code.

The short bar to the left of the SPACE bar is labeled LINE FEED and depression of this bar will cause the typewriter carriage to move the paper up one line (feed a line of paper).

To the right of the SPACE bar is a short bar labeled SPECIAL. This bar is used in conjunction with the BACKSPACE key and the x key.

When the SPECIAL bar is depressed, depression of the BACKSPACE key will backspace the typewriter carriage and the tape. It is now possible to depress the x key with the SPECIAL bar depressed and a code delete will be punched over the character on the tape and an x imprinted over the character on the print-out which was deleted from the tape.

The only way to obtain a code delete is by depressing the x key while the SPECIAL bar is depressed.

The BACKSPACE key and the x key have the usual effect if the SPECIAL bar is not used.

If the RPC-4480 Typewriter is being used for input off-line, the TYPEWRITER SELECT (RPL) button must be depressed, the READER SELECT (RPL) button must be raised and START READ (RPL) must be depressed. The RPC emblem will be lighted just as in on-line use of the typewriter.

Information may now be typed (and, presumably will be punched). If the SINGLE CHARACTER Mode button (RPL) is depressed, the START READ button (RPL) must be depressed before any (or each) character is typed.

If the CONDITIONAL STOP button is depressed, all stop codes will be ignored and the off-line typewriter will remain connected to the punch until STOP READ (RPL) is depressed.

If the CONDITIONAL STOP button (RPL) is raised, the typewriter will

accept input as defined by the setting of the SINGLE CHARACTER MODE button (RPL).

If SINGLE CHARACTER MODE (RPL) is raised, then the connection between the typewriter and the selected output device is dependent entirely upon the CONDITIONAL STOP button (RPL) as described above.

THE AUXILIARY TYPEWRITER

On the left control panel of the RPC-4430 there are buttons to select an auxiliary typewriter for input and output. Similarly there are buttons on the RPC-4431 unit to select an auxiliary typewriter. These buttons do not select the same typewriter nor do the buttons on the RPC-4430 select the typewriter used on or off-line with the RPC-4600 Auxiliary Tape/Typewriter System.

An auxiliary typewriter is auxiliary to the tape/typewriter system which will select it manually for on-line use. An auxiliary typewriter will not punch or print off-line. It is an auxiliary typewriter for use by the computer while the tape/typewriter with which it is associated is being used off-line. An auxiliary typewriter might also be used in some remote location to provide access to the RPC-4000 system.

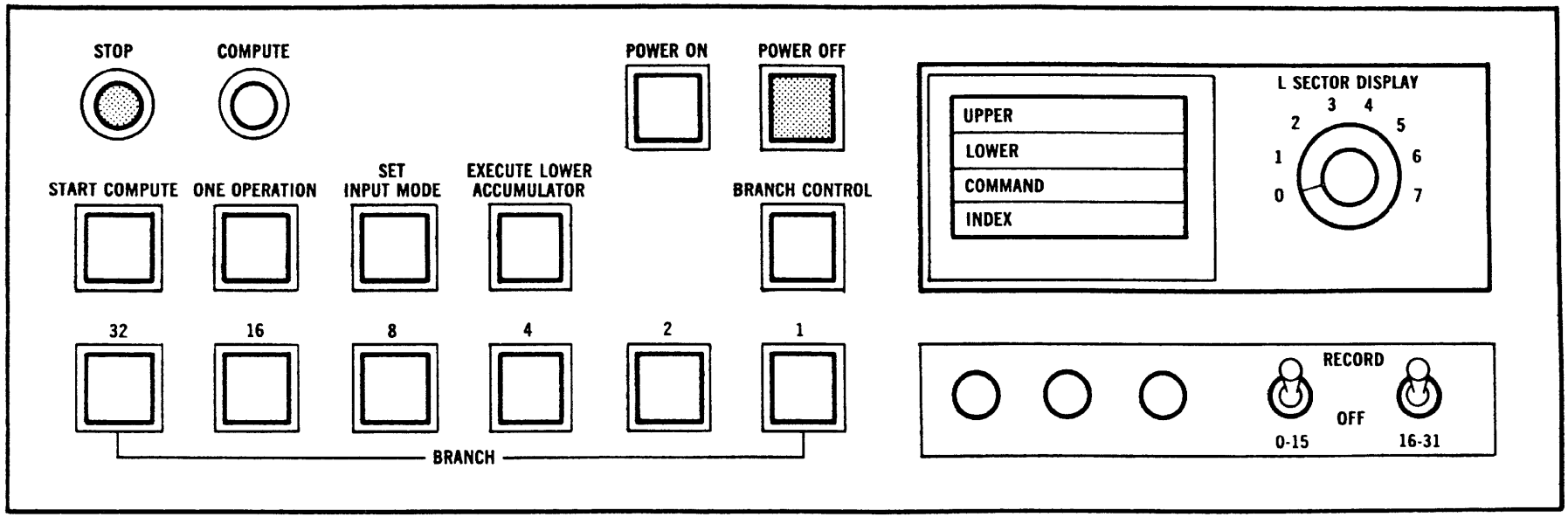


FIGURE 19. RPC-4010 Control Console

RPC-4010 CONSOLE

BUTTON OR INDICATOR	FUNCTION
POWER ON	A momentary switch which turns power ON, and is lighted when power is ON.
POWER OFF	A momentary switch which turns power OFF.
START COMPUTE	A momentary switch which initiates computer operation, depending on the setting of ONE OPERATION. It is lighted when power is ON.
ONE OPERATION	A two position switch, active when depressed and lighted when active. A depressed ONE OPERATION button places the computer in One Operation Mode. When ONE OPERATION is raised the computer is in Normal Mode. When in Normal Mode, the depression of ONE OPERATION will halt the computer at the beginning of the next phase 3 that is encountered. The depression of ONE OPERATION will allow the activation of the SET INPUT and EXECUTE LOWER ACCUMULATOR buttons. In One Operation Mode a start signal will cause the instruction word in the COMMAND Register to be executed and the instruction designated by the next-address to be placed in the COMMAND Register before the computer halts at the beginning of the next phase 3 that is encountered. See the discussion of TMI, TBC, and the Repeat Mode on page 25.
SET INPUT	A momentary switch active only when ONE OPERATION is depressed. When ONE OPERATION is active, depression of the SET

SET INPUT
(continued)

INPUT button will cause the LOWER Accumulator and the data-track field of the COMMAND Register to be set to zero. The LOWER Accumulator will be set to one-word length and the execution of a four-bit input order that overrides the instruction in the COMMAND Register will be initiated.

If SET INPUT is depressed while the computer is in One Operation Mode, the resulting four-bit input order will be in effect even though ONE OPERATION is raised before the receipt of a start signal. In all cases the successful execution of the four-bit input order is contingent on manual selection of an input device and depression of START COMPUTE.

EXECUTE LOWER ACCUMULATOR

A two position switch, active only when the ONE OPERATION button is depressed and lighted when active. When active and upon the receipt of any start signal, either as a result of the depression of START COMPUTE or of some input function, the word contained in the LOWER Accumulator is transferred into the COMMAND REGISTER. START COMPUTE must be depressed before the instruction will be executed.

Good procedure demands that EXECUTE LOWER ACCUMULATOR be raised during operation in Normal Mode. If ONE OPERATION is depressed with the operator unaware of the fact that EXECUTE LOWER ACCUMULATOR is depressed much confusion can result.

BRANCH CONTROL

A momentary switch used to turn Branch Control OFF when the computer is in One Operation Mode. This button is lighted when Branch Control is ON regardless of mode.

If the operator is single-stepping through a program in One Operation Mode, this button allows a choice of

	the path that will be taken by TBC instructions.
SENSE SWITCHES	Six two position switches used in conjunction with the SNS instructions. Each button is lighted when depressed. Depressed buttons are ON, raised buttons are OFF. (See page
OSCILLOSCOPE	Displays a visual representation of the contents, in binary, of the UPPER, LOWER, COMMAND and INDEX Register.
L-DISPLAY	An eight-position rotary switch, the setting of which determines the LOWER Accumulator that will be displayed when the LOWER is in the Eight-word mode.
STOP	An indicator lamp that is lighted when the computer is halted on HLT, INP, or as a result of manual intervention by depression of the ONE OPERATION button.
COMPUTE	An indicator lamp, lighted when the computer is executing instructions.

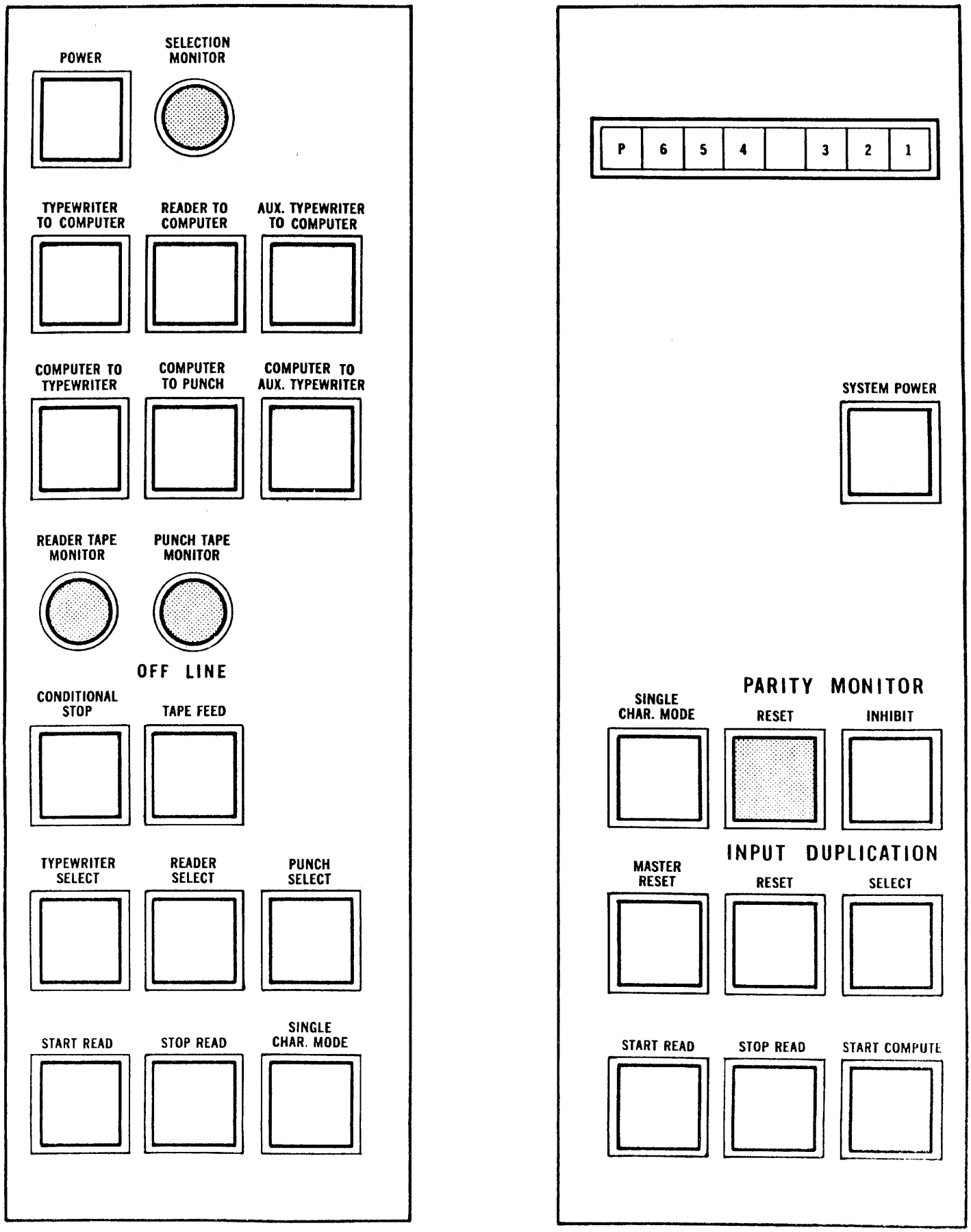


FIGURE 20. RPC-4430 Reader/Punch Control Panels

RPC-4430 READER/PUNCH RIGHT CONTROL PANEL

BUTTON OR INDICATOR	FUNCTION
SYSTEM POWER	A two-position switch, active when depressed and lighted when active. The button controls power to all input-output devices in the system. Individual power on switches cannot be activated unless SYSTEM POWER is ON.
SINGLE CHARACTER MODE	A two-position switch, active when depressed and lighted when active. When SINGLE CHARACTER MODE is depressed, all legal tape characters enter the accumulator and input will be one character at a time, with a start signal sent to the computer after each character is read. When raised, only tape codes 16 through 62 enter the computer and a start signal is sent only when a stop code is detected by the selected input device or START COMPUTE is depressed. SINGLE CHARACTER MODE cannot be controlled by programming.
PARITY MONITOR INHIBIT	A two-position switch, active when depressed and lighted when active. When active, the parity checking circuitry is disabled. When raised, the parity checking circuitry is active.
PARITY MONITOR RESET	A momentary switch. When the PARITY MONITOR INHIBIT button is raised and a character with bad parity (an uneven number of bits in all 7 channels) is under the read-head, the button is lighted. Depress the button and parity checking is by-passed for the character being read. (See also page 145).
MASTER RESET	A momentary switch which, when depressed, deselects any input/output units selected on-line. The button is lighted when the RPC-4430 power is on.

INPUT DUPLICATION SELECT	A momentary switch lighted when active. When depressed, the button turns on Copy Mode whereby all selected output devices duplicate the input data as it is read into the computer.
INPUT DUPLICATION RESET	A momentary switch which, when depressed, turns OFF Copy Mode. Copy Mode can be controlled by programming.
START READ	A momentary switch which, when depressed, activates the input device that was selected manually after the initiation of the programmed INP command or allows completion of an INP command that was interrupted by depression of STOP READ (RPR).
STOP READ	A momentary switch used to stop input by the selected input device. START READ (and only START READ) must be depressed to resume input without interrupting the input sequence. The button is lighted when the RPC-4430 power is on.
START COMPUTE	A momentary switch initiating program execution by the computer. Depressing this button is the same as depressing the START COMPUTE button on the computer console. The button is lighted when the RPC-4430 power is on.
CHARACTER INDICATOR LIGHTS	These lights represent the bit-pattern of the tape character being read by the RPC-4430. They are lighted to show the character code of the next character to be read into the system by the tape reader.

RPC-4430 READER/PUNCH LEFT CONTROL PANEL

BUTTON OR INDICATOR	FUNCTION
POWER ON	A two position switch used to control the electrical power of the RPC-4500. It is active when depressed and lighted when active. POWER ON should be raised and left raised until SYSTEM POWER is on.
SELECTION MONITOR	An indicator lamp that is lighted when an I/O device selected off-line is selected by the computer, manually or by a program step. (See page 123).
TYPEWRITER TO COMPUTER	A momentary switch to select the typewriter for input. The button is lighted when the typewriter is selected on-line. *
READER TO COMPUTER	A momentary switch to select the reader for input. The button is lighted when the reader is selected on-line. *
AUXILIARY TYPEWRITER TO COMPUTER	A momentary switch identical to TYPEWRITER TO COMPUTER for the auxiliary typewriter of the RPC-4430.
COMPUTER TO TYPEWRITER	A momentary switch to select the typewriter for output. It is lighted when the typewriter is selected on-line. *
COMPUTER TO PUNCH	A momentary switch to select the punch for output. It is lighted to indicate the punch is selected on-line. *

* These buttons will be lighted when the related device is selected on-line manually or by a program step. The computer does not have access to the device if it is also selected off-line. (See SELECTION MONITOR (above) and the discussion on page 123.

COMPUTER TO AUXILIARY TYPEWRITER	A momentary switch identical to COMPUTER TO TYPEWRITER for the auxiliary typewriter of the RPC-4430.
READER TAPE MONITOR	Two interlock switches will turn on this indicator. One is a pressure switch under the reading head that will stop the reader and turn on the indicator if the reader is out of tape. The other is a pressure switch at the point where the tape feeds over the edge of the cabinet. When tension on the tape as it feeds toward the reader becomes too great the switch will stop the reader and turn on the indicator. The appropriate STOP READ button should be depressed before corrective action is taken.
PUNCH TAPE MONITOR	An indicator light, lighted when the punch is out of tape or is jammed. Requires immediate attention at any time.
CONDITIONAL STOP	A two-position switch, active when depressed and lighted when active. When inactive (raised), a stop code sensed by any selected input device will terminate off-line operation. Operation must be then resumed by depressing START READ (RPL). When active (depressed), stop codes have the same effect as other characters. Operation will continue until STOP READ (RPL) is depressed. Activation of CONDITIONAL STOP will override SINGLE CHARACTER MODE (RPL), if it is active.
TAPE FEED	A momentary switch which, when the punch is selected off-line, will feed blank tape with sprocket holes for leading and trailing ends of a tape. The button is lighted when RPC-4430 power is on.
TYPEWRITER SELECT	A two-position switch, active when depressed and lighted when active, that will de-select the typewriter if it is selected on-line and select it off-line; for output if another input device is selected off-line, or for input if no other device is selected off-line.

READER SELECT	A two-position switch, active when depressed and lighted when active, that will de-select the reader if it is selected on-line and select it off-line. While the reader is selected off-line, the typewriter may not be used for off-line input.
PUNCH SELECT	A two-position switch, active when depressed and lighted when active, that will de-select the punch if it is selected on-line and select it off-line.
START READ	A momentary switch which, when depressed, connects the device selected for off-line input to the device selected for off-line output and otherwise activates the input device. The START READ signal is active as specified by SINGLE CHARACTER MODE (RPL) or CONDITIONAL STOP (RPL). The button is lighted when RPC-4430 power is on.
STOP READ	A momentary switch which, when depressed, terminates off-line input from the selected input device. The button is lighted when RPC-4430 power is on.
SINGLE CHARACTER MODE	A two-position switch, active when depressed and lighted when active. When depressed, off-line input will be character by character. If CONDITIONAL STOP is depressed, SINGLE CHARACTER MODE will have no effect, even though it will appear to be active.

BOOTSTRAP

Bootstrapping is the process of entering a sufficient number of program steps, which in harmony with manual operations, will allow the entry of a short program loop that will load instruction words into the computer memory. There is often some confusion in the use of the term bootstrap. For this discussion, the terms bootstrap, load program and input program will be used.

An input program is distinct from bootstrap or loading programs in that the input program processes the information in some way, such as scaling, binarization, conversion or a combination of these operations. The load program is a complete program, in that the entry of information is independent of manual intervention, so we may speak of the computer as being "under program control".

The bootstrap is not separable from manual intervention. Part of a bootstrap may be free of manual intervention but the computer is not considered under program control until the program is free of manual control and has reached the point where the program ceases to modify itself through input. Once the program is in memory and instruction words are being loaded in some assigned location by a program loop that is not being changed as a result of that input, we are out of the bootstrap and under control of the load program.

Bootstrapping for the RPC-4000 involves several manual operations but it is very simple once the procedure has been used.

The bootstrapping operation consists of the following:

Depress READER TO COMPUTER (RPL)

Depress ONE OPERATION (CC)

Depress SET INPUT (CC)

Depress EXECUTE LOWER ACCUMULATOR (CC)

Depress START COMPUTE (CC) (RPR)

Remembering that we are in One Operation Mode, we see that information will be input until a stop code is detected by the reader. At this point, the reader stops and a start signal is given to the computer. EXECUTE LOWER ACCUMULATOR is active so the computer, upon receipt of the start signal, transfers the contents of the LOWER to the COMMAND Register and halts with the instruction word that was input now in the COMMAND Register as the next instruction to be executed.

Depress START COMPUTE (CC) (RPR)

The instruction in the COMMAND Register is executed.

If the tape contained DFFOBF84D7F03F80*, we would recognize this as a hexadecimal tape word of 16 characters consisting of two computer words of 8 characters each. The equivalent ROAR language words would be:

```
CLL 12702    12702
CLU 12700    12700
```

Since this word is now in the double-length accumulator with the instruction word in the LOWER just executed, let us examine the execution that has taken place.

ROAR language will be used in the discussion, but it must be remembered that the bootstrap tape is a hexadecimal tape.

The instruction we placed in the COMMAND Register was:

```
CLU 12700    12700
```

The double-length accumulator contained:

```
UPPER        CLL 12702    12702
LOWER        CLU 12700    12700
```

When the instruction in the COMMAND Register was executed by depressing START COMPUTE (CC) (RPR), the contents of the UPPER was transferred to 12700.

The next-address specified is 12700. In memory, we have:

```
12700        CLL 12702    12702
```

NOW:

```
Raise EXECUTE LOWER ACCUMULATOR     (CC)

Depress SET INPUT                   (CC)

Raise ONE OPERATION                 (CC)

Depress START COMPUTE               (CC) (RPR)
```

When START COMPUTE (CC) (RPR) is depressed, the four-bit input order is executed and the double-length accumulator contains:

```
UPPER        INP      0    12700
LOWER        CLU  12707    12707
```

The computer is in Normal Mode, so when the stop code is detected by the reader, a start signal is sent to the computer.

The next instruction to be executed is specified by the next-address of the COMMAND Register which is:

```
12700        CLL          12702   12702
```

The instruction is executed and upon execution, memory looks like:

```
12700        CLL          12702   12702
12702        CLU          12707   12707
```

and we proceed:

```
12702        CLU          12707   12707
```

The UPPER is cleared by the execution, so that we have in memory:

```
12700        CLL          12702   12702
12702        CLU          12707   12707
12707        INP              0   12700
```

On the execution of the instruction in 12707, the double-length accumulator will receive two additional words:

```
UPPER        INSTRUCTION WORD
LOWER        CLU       X          12707
```

The UPPER will contain the instruction word to be stored in the location specified by the data-address of the word in the LOWER.

This sequence is repeated until the load program has been stored in memory.

```
UPPER        INSTRUCTION WORD
LOWER  CLU        X      LOAD
```

where X in the last location in the load program and LOAD is the beginning of the load program.

The load program should meet the following requirements:

a. input a tape word
b. determine where the word is to be stored
c. perform that storage
d. check for completion of input

    1. if complete, transfer to routine
    2. if not complete, loop to read next tape word

e. the load program might also checksum the tape or audit the accuracy of the input device in some way.

The following is a short routine which will read and store a hexadecimal tape with the same word format as the ROAR output tape except that this routine does not checksum and the final tape word must be stored in LOAD for transfer to the routine.

```
LOAD    INP     0     (A)  READ 1 TAPE WORD
(A)     SRL     114   (B)  LOC. OF WORD TO D-ADDRESS
(B)     EXC     498   (C)  ADDRESS TO X
(C)     SRL     14    (D)  RESTORE TAPE WORD
(D)     XCLL    0     LOAD STORE IN PROPER LOC.
```

OPERATING PROCEDURES

1. RPC-4000 STARTING PROCEDURE
(Assume power off, all units)

1.1. RPC-4430 Reader/Punch

1.1.1. Make certain POWER ON (RPL) is raised.
1.1.2. Depress SYSTEM POWER (RPR)
1.1.3. Depress POWER ON (RPL)

1.2. RPC-4010 Computer Console

1.2.1. Depress POWER ON
1.2.2. Raise all SENSE SWITCHES

NOTE: Approximately three minutes are required for the drum to reach the required speed.

2. BOOTSTRAPPING PROCEDURE

2.1. Depress MASTER RESET (RPR) to deselect all I/O devices selected on-line.

2.2. Deselect (by raising the appropriate button) any device selected off-line. (RPL)
(If a device is selected off-line it is not available to the RPC-4010.)

2.3. The following steps read the bootstrap from tape:
Place the tape in the reader.

2.3.1. Depress READER TO COMPUTER (RPL)
2.3.2. Depress ONE OPERATION (CC)
2.3.3. Depress SET INPUT (CC)
2.3.4. Depress EXECUTE LOWER ACCUMULATOR (CC)
2.3.5. Depress START COMPUTE (CC) (RPR)
2.3.6. WAIT for reader to stop
2.3.7. Depress START COMPUTE (CC) (RPR)
2.3.8. Raise EXECUTE LOWER ACCUMULATOR (CC)
2.3.9. Depress SET INPUT (CC)

2.3.10. Raise ONE OPERATION (CC)
2.3.11. Depress START COMPUTE (CC) (RPR)
2.3.12. The bootstrap will now read in and begin loading the hexadecimal program tape without stopping.

3. MANUAL ENTRANCE TO A PROGRAM

3.1. Depress MASTER RESET (RPR) to deselect all I/O devices selected on-line.
3.2. Deselect any device selected off-line. (RPL)
3.3. Depress ONE OPERATION (CC)
3.4. Depress SET INPUT (CC)
3.5. Depress EXECUTE LOWER ACCUMULATOR (CC)
3.6. Depress TYPEWRITER TO COMPUTER (RPL)
3.7. Depress START COMPUTE (CC) (RPR)
3.8. Type the starting address of the program, in hexadecimal, followed by a stop code "*".
3.9. Raise EXECUTE LOWER ACCUMULATOR (CC)
3.10. Raise ONE OPERATION (CC)
3.11. Depress START COMPUTE (CC) (RPR)

4. ERRORS IN LOADING MASTER TAPES

The two possible errors (other than manual errors) are checksum and parity errors.

4.1. Checksum Errors

If a checksum error is encountered, the typewriter prints "ERROR" and the computer halts. Recovery from checksum errors must be made by the procedure outlined below.

4.1.1. Remove the tape from the reader.
4.1.2. Position the tape to the space between the checksum of the record in which the error was detected and the beginning of the last record read correctly.
4.1.3. Depress START COMPUTE (RPR) (CC)
4.1.4. Try the procedure at least three times. If the third attempt to read the record is unsuccessful, go to 5.5.

4.2. Parity Errors

When a parity error occurs, the light under the PARITY MONITOR RESET button (RPR) lights up and the computer stops.

4.2.1. Depress STOP READ (RPR)
4.2.2. Check the tape character
4.2.3. Depress PARITY MONITOR RESET (RPR)
4.2.4. Depress START READ (RPR)
4.2.5. If the computer does not stop on a checksum error at the end of the record, the parity was false and the character was correct. Checksums are the most reliable checks.
4.2.6. If a checksum error was detected, follow the procedure 4.1.3. (See page 123 if any difficulty).

NOTE: Generally one tape word will be read from the next record before the checksum error is detected.

5. USE OF ROAR TO ASSEMBLE PROGRAMS

5.1. Set SENSE SWITCHES (See ROAR Manual)
5.2. Place the ROAR input tape in the reader.

5.2.1. Depress MASTER RESET (RPR)
5.2.2. Deselect any equipment selected off-line (RPL)
5.2.3. Depress READER TO COMPUTER (RPL)
5.2.4. Depress COMPUTER TO TYPEWRITER (RPL)
5.2.5. If the computer is stopped on a HLT command, depress START COMPUTE (CC) (RPR)
5.2.6. If the computer is stopped on INP command, depress START READ (RPR)
5.2.7. ROAR will read the input tape and assemble the program.

5.3. Errors during ROAR assembly

The errors considered in this section are:

Input
Parity
Machine Malfunctions

5.3.1. Correction of input errors.

These errors are either punching or coding errors that ROAR will detect and log out with appropriate comment.

Follow the recovery procedure below for manual correction of these errors:

5.3.1.1. DO NOT MOVE THE INPUT TAPE
5.3.1.2. With the computer halted, depress MASTER RESET (RPR)
5.3.1.3. Depress TYPEWRITER TO COMPUTER (RPL)
5.3.1.4. Depress START COMPUTE (CC) (RPR)
5.3.1.5. Starting with the location of the instruction in which the error occurred, type the decimal location assigned by ROAR (or in some cases the last location assembled correctly.) Refer to the coding sheet and type the correct information up to and including the last stop code previously read by ROAR. ROAR will assemble the manual input as each stop code is entered.
5.3.1.6. Depress MASTER RESET (RPR)
5.3.1.7. Depress READER TO COMPUTER (RPL)
5.3.1.8. Depress COMPUTER TO TYPEWRITER (RPL)
5.3.1.9. Depress START READ (RPR)
5.3.1.10. ROAR will resume assembly.

5.4. Parity Errors

5.4.1. Depress STOP READ (RPR)
5.4.2. Mark the tape so the character causing the parity stop will be easy to remember. DO NOT REMOVE THE TAPE. (See also 6).
5.4.3. Depress PARITY MONITOR RESET (RPR)
5.4.4. Depress TYPEWRITER TO COMPUTER (RPL)
5.4.5. Type any six hexadecimal characters, other than zero, followed by a stop code to cause an error print-out.
5.4.6. Depress MASTER RESET (RPR)
5.4.7. Depress READER SELECT off-line (RPL)
5.4.8. Make sure CONDITIONAL STOP (RPL) is raised.
5.4.9. Depress START READ (RPL)
The tape will now be positioned at the first character of the field immediately following the field where the parity occurred.

5.4.10. Raise READER SELECT (RPL). (Raise buttons depressed 5.4.7. through 5.4.8.)
5.4.11. Refer to coding sheet and follow procedure in 5.3.1.1. through 5.3.1.10.

5.5. Machine Malfunctions

It is generally best to try the procedures outlined above. If after several tries, recovery proves impossible, call your friendly service man.

6. If it is desirable to correct a tape off-line, this may be done by deselecting the Reader and leaving ROAR intact in the computer with its output undisturbed. In this case, the recovery would be by means of the following steps.

Place the corrected tape in the Reader and position it to the beginning of the last non-blank location assembled by ROAR and follow steps 5.2.1. through 5.2.7.

7. USE OF THE RPC-4500 TAPE-TYPEWRITER SYSTEM OFF-LINE

To use the RPC-4500 to punch tapes off-line, the following sequence should be followed:

7.1. Depress MASTER RESET (RPR)
(Deselects all on-line units)
7.2. Depress TYPEWRITER SELECT (RPL)
7.3. Depress PUNCH SELECT (RPL)
7.4. Depress CONDITIONAL STOP (RPL)
7.5. Depress START READ (RPL)
7.6. Whatever is typed will be punched

8. POWER OFF PROCEDURE

When the computer is not to be used for a considerable time or the installation is closing shop for the day, the following procedure is recommended for powering down.

The sequence will leave the system in a condition that will make it difficult for the next user to make mistakes in selection and switch settings.

8.1. Raise EXECUTE LOWER ACCUMULATOR (CC)
8.2. Depress ONE OPERATION (CC)
8.3. Raise all SENSE SWITCHES (CC)
8.4. Depress POWER OFF (CC)
8.5. Depress MASTER RESET (RPR)
8.6. Deselect any I/O devices in the system that are selected off-line.
8.7. Raise POWER ON buttons (RPL, etc.) of the various I/O devices in the system.
8.8. Raise SYSTEM POWER (RPR)
8.9. The RPC-4000 ELECTRONIC COMPUTING SYSTEM is now in the best condition for use by the next person who has need of its capabilities.

7 SUMMARY LISTS AND TABLES

ALGEBRAIC EXPRESSION OF THE RPC-4000 COMMANDS

This section contains a description, in algebraic notation, of the internal result of execution of RPC-4000 commands. The symbols used are defined on the following pages. The methods used in the notation and the assumptions on which the algebraic descriptions are based are discussed.

DEFINITION OF SYMBOLS

t will mean the contents of the data-track portion (bits 5 thru 11) of a register or location.

s will mean the contents of the data-sector portion (bits 12 thru 17) of a register or location.

d will mean the data-address field (bits 5 thru 17) of a register or location. d will always mean that the data-track (t) and sector (s) fields are each of significance either as track and sector of an address or that the track and/or sector are used to modify the execution of an instruction.

n will mean the next-address field (bits 18 thru 30) of a register or location.

i will mean the index indicator bit (bit 31) of a register or location.

c will mean "contents of" the bit positions or location denoted by the symbol modified by c. The notation c(d) would mean the contents of bits 5 thru 17 of a register or location, c(U) would mean the contents of the UPPER Accumulator.

D will be used to denote the contents of bits 5 thru 17 (the data-address field) of a register or location when these bits specify some memory location.

N will mean the contents of bits 18 thru 30 (the next-address field) of a register or location when these bits specify a location in memory.

C will mean the COMMAND Register.

C' will mean the command field (bits 0 thru 4) of a register or location. The contents of the command field [c(C')]will generally be expressed as the mnemonic code for a given command.

v will mean bits 5 thru 17 of a register or location when these bits contain some value to be used by the instruction word. (See LDX and CXE, pages 30 and 36.)

With the above terms we can express any instruction word. Statements concerning an instruction word assume that the instruction exists in the COMMAND Register. The contents of a COMMAND Register can be expressed as the c(C'), c(d), c(n), and the c(i). More properly, the assumption is made that the COMMAND Register will contain:

(Mnemonic code) D N

Certain other symbols are needed to denote other bit positions and locations of significance. These symbols identify portions of registers and locations acted upon by instructions or describe some unique effect of the execution of an instruction.

U will mean the UPPER Accumulator, c(U) will mean the contents of the UPPER Accumulator.

L will mean the LOWER Accumulator. c(L) will mean the contents of the LOWER Accumulator.

e will mean some one of the eight words in the Eight-word LOWER Accumulator, e will be equal to 0, 1, 2, ------- or 7. e will never be greater than 7.

L_e will designate the Eight-word LOWER Accumulator where e is equal to the modulo 8 equivalent of the data-sector of the effective operand address. (See the discussion page 24).

X will mean the INDEX Register. c(X) will mean the contents of the INDEX Register.

r will mean bits 18 thru 24 (the Repeat Count) of a register or location. Thus $C(D_r)$ will mean the contents of bits 18 thru 24 of the location specified by D.

T will mean the Sector Reference Timing Track. c(T) will be the contents of the Sector Reference Timing Track.(See page 20.)

f will mean bits 25 thru 30 of a register or location. f will generally refer to those bits holding the "Found Sector". (See also pages 50 and 51).

The operators used to describe the effect of the execution of an instruction are as follows:

+	will mean addition.
-	will mean subtraction
x	will mean multiplication.
$\frac{a}{b}$	will mean division.
=	will mean equal.
$\neq$	will mean not equal.
<	will mean less than.
>	will mean greater than.
$\leq$	will mean less than or equal to.
$\geq$	will mean greater than or equal to.
:	will mean therefore.
$\longrightarrow$	will mean replaces.
$\not\longrightarrow$	will mean replacement terminates.

These are generally arithmetic operators; no attempt is made in this section to describe logical addition, etc. as a result of the execution of an instruction.

BC will mean the Branch Control flip-flop. Branch Control is assumed to be <u>on</u> unless otherwise specified.

$\overline{BC}$ means not Branch Control or Branch Control <u>off</u>.

The expression $\overline{BC} \longrightarrow BC$ will mean Branch Control off replaces Branch Control regardless of its state; $BC \longrightarrow BC$ will mean Branch Control on replaces Branch control regardless of its state.

Modifiers are necessary in some expressions. The problem of describing which bits in which registers are affected is a necessity for those instructions that affect only part of a register or location. An LDC instruction acts only on bits 18 thru 24 of the operand address and the INDEX Register; other instructions such as CXE, LDX, and SAU, pose similar problems.

Subscripts are used to show which bits in a given register or location are affected (or used). X_r would mean only bits 18 thru 24 of the INDEX Register are used/affected by an instruction. D_d would mean only bits 5 thru 17 of the location specified by the data-address are used/affected by the execution of an instruction.

Similarly any lower case symbol may be used as a subscript. The subscript means that only those bit positions designated by the lower case symbol are used/affected or are of special significance such as determining the LOWER Accumulator affected when the LOWER is 8-words.

The use of masks will be shown as follows:

m will be used as a prefix to denote that the effect of an instruction's execution is determined by a mask in the location designated.

mu means the contents of the UPPER is used as a mask.

ml means the contents of the LOWER is used as a mask.

md' means the contents of the location specified by the data-address is used as a mask.

Thus c(mlU) means the contents of the UPPER modified (specified) by the mask in the LOWER, c(md'U) means the contents of the UPPER modified (specified) by the mask in the location specified by the data-address.

The symbols above will describe most instructions executed in the Normal Mode. The COMMAND Register contains an instruction word. The search for sector begins and then:

$$c(N) \longrightarrow c(C)$$

Any sequence of instructions can be represented by:

$$c(N) \longrightarrow c(C) \longrightarrow c(N_1) \longrightarrow c(C)$$
$$c(N_2) \longrightarrow c(C) \longrightarrow \text{etc.}$$

THE INDEXING EQUATION

$i = 1$

$$c(N_d) + c(X_v) \longrightarrow c(C_d)$$

as opposed to

$i = 0$

$$c(N_d) \longrightarrow c(C_d)$$

REPEAT MODE

The symbols as explained above would allow us to describe the execution of any instruction in the Normal Mode. There will be many instances when an instruction will be executed in the Repeat Mode. Several terms are necessary to describe that phenomena.

R will mean Repeat Mode

$\overline{R}$ will mean Normal Mode

j will be the momentary value by which the sector portion of the effective operand address is incremented during execution of an instruction in the Repeat Mode. At the first execution j will equal 0, on the second j will equal 1 and the final execution j will equal the value of the Repeat Count loaded by the LDC instruction initiating the Repeat Mode. (See page 25).

By "effective operand address" is meant the data-address used for any one execution of an instruction. If the instruction is not indexed and is being executed in the Normal Mode the effective operand address and the data-address are the same. If an instruction is indexed, the data-address of the instruction word is incremented by the index value and the sum is placed in the COMMAND Register as the data-address. We are concerned here with the three different "data-address fields", but only one effective operand address.

If an instruction is being executed in the Repeat Mode, the data-address (which may be indexed) is placed in the COMMAND Register and remains unchanged. However, the instruction is executed once and then the effective operand address is incremented by one (COMMAND Register unchanged), executed again, the effective operand address is incremented by one again, etc. This sequence is repeated until the effective operand address is equal to the data-address (D) of the COMMAND Register plus the value (r) loaded by the LDC instruction word initiating the Repeat Mode.

The statement of an instruction where R = R is concerned with each individual execution, not with all of the executions which transpire. The expression D_{s+j} will mean that the sector portion of D is incremented by the momentary value of j to determine which sector is affected by (or is used by) that particular execution. This is extremely important when the instruction uses/affects the Eight-word LOWER.

To illustrate, RAL repeated with one-word LOWER requires knowledge about the final execution and no other; the use of RAL repeated with an Eight-word LOWER requires knowledge about each and every execution. The latter is the usual case; we are most often concerned with each execution or a particular execution, but not with the final result of all the executions.

FINDING L_e

L_e is determined by the effective operand address of the instruction that uses/affects the Eight-word LOWER.

In Normal Mode,

$$e = \text{the remainder of } \frac{C_s}{8}$$

In Repeat Mode, e is obtained by:

$$e = \text{the remainder of } \frac{C_s + j}{8}$$

DESCRIPTION OF COMMAND EXECUTION

(All numeric values are decimal.)

HLT d N

t must equal 0

The computer stops.

SNS d N

$0 < t < 128$

Initially $\overline{BC} \longrightarrow BC$, then if any bit in t that is equal to 1 corresponds to any depressed SENSE SWITCH, $BC \longrightarrow BC$; otherwise $\overline{BC} \longrightarrow BC$.

CXE v N

$\overline{BC} \longrightarrow BC$; if $c(X_v) = c(C_v)$: $BC \longrightarrow BC$

$\overline{BC} \longrightarrow BC$; if $c(X_v) \neq c(C_v)$: $\overline{BC} \longrightarrow BC$

NOTE: xCXE (v = 1) N BC guaranteed OFF

xCXE (v = 0) N BC guaranteed ON

RAU D N

$R = \overline{R}$

$c(D) \longrightarrow c(U)$

$R = R$

$c(D_{s+j}) \longrightarrow c(U)$

RAL D N

$R = \overline{R}, \quad L = L$

$c(D) \longrightarrow c(L)$

$R = \overline{R}, \quad L = L_e$

$c(D_s) \longrightarrow c(L_e)$

$R = R, \quad L = L$

$c(D_{s+j}) \longrightarrow c(L)$

$R = R, \quad L = L_e$

$c(D_{s+j}) \longrightarrow c(L_e)$

SAU D N

$R = \overline{R}$ $\quad c(U_d) \longrightarrow c(D_d)$

$R = R$ $\quad c(U_d) \longrightarrow c(D_{(s+j)d})$

MST D N

$R = \bar{R}$, $L = L$

$c(muL) \longrightarrow c(muD)$

$R = \bar{R}$, $L = L_e$

$c(muL_e) \longrightarrow c(muD_s)$

$R = R$, $L = L$

$c(muL) \longrightarrow c(muD_{s+j})$

$R = R$, $L = L_e$

$c(muL_e) \longrightarrow c(muD_{s+j})$

LDC D N

$R = \bar{R}$

$\bar{R} \longrightarrow R$, $c(D_r) \longrightarrow c(X_r)$, $R \longrightarrow R$

$R = R$ Is an impossibility.

LDX v N

$R = \bar{R}$, $c(C_i) = 0$

$c(C_v) \longrightarrow c(X_v)$

$R = \bar{R}$, $c(C_i) = 1$,

$c(C_v) + c(X_v) \longrightarrow c(X_v)$

$R = R$ No difference.

INP	d	N	Not considered.
EXC	d	N	
	s	NOT USED	
	$R = \bar{R}$,	$L = L$	
		$t = 0$	
			NO OPERATION
		$t = 1$	
			$c(U) \longrightarrow c(L)$
		$t = 2$	
			$c(L) \longrightarrow c(U)$
		$t = 4$	
			$c(U) \longrightarrow c(X)$
		$t = 8$	
			$c(X) \longrightarrow c(U)$
		$t = 16$	
			$L_e \longrightarrow L$
		$t = 32$	
			$L \longrightarrow L_e$
	$R = \bar{R}$,	$L = L_e$	NOT CONSIDERED
	$R = R$,	$L = L$	
	$R = R$,	$L = L_e$	

DVU D N

$R = \overline{R}$, $L = L$

$\frac{c(U)}{c(D)} \longrightarrow c(U)$, positive remainder of

$\frac{c(U)}{c(D)} \longrightarrow c(L)$

$R = \overline{R}$, $L = L_e$

$R = R$, $L = L$ } Not considered.

$R = R$, $L = L_e$

DIV D N

$R = \overline{R}$, $L = L$

$\frac{c(UL)}{c(D)} \longrightarrow c(U)$, positive remainder of

$\frac{c(UL)}{c(D)} \longrightarrow c(L)$

$R = \overline{R}$, $L = L_e$

$R = R$, $L = L$ } Not considered.

$R = R$, $L = L_e$

SLC d N

A normalized number in the double-length accumulator is a number whose most significant digit (bit) is in bit 1.

$R = \overline{R}$, $L = L$

normalized $c(UL) \longrightarrow c(U)$, $0 \longrightarrow c(L)$, number of shifts + $c(C_s)$ modulo $64 \longrightarrow c(L_s)$.

NOTE: If the number of shifts plus $c(C_s)$ equals 64, shifting stops and $0 \longrightarrow c(L)$. The above is true unless the c(UL) is a normalized number. If c(UL) is normalized then $c(C_s) \longrightarrow c(L_s)$; if c(UL) = 0 then number of shifts will equal 64 - $c(C_s)$ and $0 \longrightarrow c(L_s)$

$R = \overline{R}$,	$L = L_e$	Not considered.
$R = R$,	$L = L$	
$R = R$,	$L = L_e$	

MPY D N

$R = \overline{R}$, $L = L$

$c(U) \times c(D) \longrightarrow c(UL)$

$R = \overline{R}$,	$L = L_e$	Not considered.
$R = R$,	$L = L$	
$R = R$,	$L = L_e$	

MPT | d N

$R = \overline{R}$

$t = 0$

$c(U) \times 10 \longrightarrow c(U)$

$R = \overline{R}, \quad L = L$

$t = 64$

$c(L) \times 10 \longrightarrow c(L)$

$R = \overline{R}, \quad L = L_e$

$c(L_e) \times 10 \longrightarrow c(L_e)$

$R = R$

$t = 0$

$c(U) \times 10 \longrightarrow c(U)$

$R = R, \quad L = L$

$t = 64$

$c(L) \times 10 \longrightarrow c(L)$

$R = R, \quad L = L_e$

$t = 64$

$c(L_e \times 10 \longrightarrow c(L_e)$

Overflow is ignored.

NOTE: It is specified that these descriptions are concerned with each single execution. In this case we may be concerned with exponentiation so that a repeated MPT could be considered as

$$c(U \text{ or } L) \times 10^j \longrightarrow c(U \text{ or } L).$$

In the case $L = L_e$, $R = R$

if $j > 7$ exponentiation would take piace for those words of the Eight-word LOWER that were acted upon more than once.

PRD D N

NOT CONSIDERED

PRU D N

NOT CONSIDERED

EXT D N

$c(D) = md'$

$R = \bar{R}$

$c(md'U) \longrightarrow c(U)$

$R = R$

$c(md'_{s+j}U) \longrightarrow c(U)$

MML D N

$R = \bar{R}, \quad L = L$

$c(muD) \longrightarrow c(muL)$

$R = \bar{R}, \quad L = L_e$

$c(muD_s) \longrightarrow c(muL_e)$

$R = R, \quad L = L$

$c(muD_{s+j}) \longrightarrow c(muL)$

$R = R, \quad L = L_e$

$c(muD_{s+j}) \longrightarrow c(muL_e)$

CME D N

$R = \overline{R}$, $L = L$

$\overline{BC} \longrightarrow BC$ then if

$c(m1U) = c(m1D)$: $BC \longrightarrow BC$,

or if $c(m1U) \neq c(m1D)$: $\overline{BC} \longrightarrow BC$.

$R = \overline{R}$, $L = L_e$

$\overline{BC} \longrightarrow BC$ then if

$c(m1_eU) = c(m1_eD_s)$: $BC \longrightarrow BC$,

or if $c(m1_eU) \neq c(m1_eD_s)$: $\overline{BC} \longrightarrow BC$.

$R = R$, $L = L$

$\overline{BC} \longrightarrow BC$, $c(T+j) \longrightarrow c(X_f)$

then if $c(m1U) = c(m1D_{s+j})$: $BC \longrightarrow BC$,

$c(T+j) \not\longrightarrow c(X_f)$;

or if $c(m1U) \neq c(m1D_{s+j})$: $\overline{BC} \longrightarrow BC$,

$c(T+j) \longrightarrow c(X_f)$.

$R = R$, $L = L_e$

$\overline{BC} \longrightarrow BC$, $c(T+j) \longrightarrow c(X_f)$,

then if $c(m1_eU) = c(m1_eD_{s+j})$: $BC \longrightarrow BC$,

$c(T+j) \not\longrightarrow c(X_f)$;

or if $c(m1_eU) \neq c(m1_eD_{s+j})$: $\overline{BC} \longrightarrow BC$,

$c(T+j) \longrightarrow c(X_f)$.

CMG D N

$R = \overline{R}$, $L = L$

$\overline{BC} \longrightarrow BC$;

then if $c(m1U) \leq c(m1D)$: $BC \longrightarrow BC$

or if $c(m1U) > c(m1D)$: $\overline{BC} \longrightarrow BC$

$R = \overline{R}$, $L = L_e$

$\overline{BC} \longrightarrow BC$

then if $c(m1_eU) \leq c(m1_eD_s)$: $BC \longrightarrow BC$

or if $c(m1_eU) > c(m1_eD_s)$: $\overline{BC} \longrightarrow BC$.

$R = R$, $L = L$

$\overline{BC} \longrightarrow BC$ $\quad c(T+j) \longrightarrow c(X_f)$,

then if $c(m1U) \leq c(m1D_{s+j})$: $BC \longrightarrow BC$,

$c(T+j) \not\longrightarrow c(X_f)$;

or if $c(m1U) > c(m1D_{s+j})$: $\overline{BC} \longrightarrow BC$,

$c(T+j) \longrightarrow c(X_f)$.

$R = R$, $L = L_e$

$\overline{BC} \longrightarrow BC$, $\quad c(T+j) \longrightarrow c(X_f)$,

then if $c(m1_eU) \leq c(m1_eD_{s+j})$: $BC \longrightarrow BC$,

$c(T+j) \not\longrightarrow c(X_f)$;

or if $c(m1_eU) > c(m1_eD_{s+j})$: $\overline{BC} \longrightarrow BC$,

$c(T+j) \longrightarrow c(X_f)$.

TMI D N

R = $\overline{R}$

Bit 0 of U = 0 (positive)

C(N) $\longrightarrow$ c(C),

Bit 0 of U = 1 (negative)

c(D) $\longrightarrow$ c(C)

R = R NOT CONSIDERED (See page 25)

TBC D N

R = $\overline{R}$

BC = $\overline{BC}$

c(N) $\longrightarrow$ c(C)

BC = BC

c(D) $\longrightarrow$ c(C) $\longrightarrow$ $\overline{BC}$ $\longrightarrow$ BC.

R = R NOT CONSIDERED (See page 25)

STU D N

R = $\overline{R}$

c(U) $\longrightarrow$ c(D)

R = R

c(U) $\longrightarrow$ c(D_{s+j})

STL D N

$R = \overline{R}$, $L = L$

$c(L) \longrightarrow c(D)$

$R = \overline{R}$, $L = L_e$

$c(L_e) \longrightarrow c(D_s)$

$R = R$, $L = L$

$c(L) \longrightarrow c(D_{s+j})$

$R = R$, $L = L_e$

$c(L_e) \longrightarrow c(D_{s+j})$

CLU D N

$R = \overline{R}$

$c(U) \longrightarrow c(D)$, $0 \longrightarrow c(U)$

$R = R$

$c(U) \longrightarrow c(D_{s+j})$, $0 \longrightarrow c(U)$

NOTE: After the initial execution, the contents of U will be 0 and hence

$0 \longrightarrow c(D_{s+j})$ when $j \neq 0$.

CLL D N

$R = \overline{R}$, $L = L$

$c(L) \longrightarrow c(D)$, $0 \longrightarrow c(L)$

$R = \overline{R}$, $L = L_e$

$c(L_e) \longrightarrow c(D_s)$, $0 \longrightarrow c(L_e)$

NOTE: After the initial execution, the contents of L will be 0 and hence,

$0 \longrightarrow c(D_{s+j})$ when $j \neq 0$

$R = R$, $L = L$

$c(L) \longrightarrow c(D_{s+j})$, $0 \longrightarrow c(L)$

$R = R$, $L = L_e$

$c(L_e) \longrightarrow c(D_{s+j})$, $0 \longrightarrow c(L_e)$

NOTE: If the Repeat Count is greater than 7 then $0 \longrightarrow c(D_{s+j})$ after $j \geq 8$.

ADU D N

$R = \overline{R}$

$$c(U) + c(D) \longrightarrow c(U)$$

$R = R$

$$c(U) + c(D_{s+j}) \longrightarrow c(U)$$

ADL D N

$R = \overline{R}, \quad L = L$

$$c(L) + c(D) \longrightarrow c(L)$$

$R = \overline{R}, \quad L = L_e$

$$c(L_e) + c(D_s) \longrightarrow c(L_e)$$

$R = R, \quad L = L$

$$c(L) + c(D_{s+j}) \longrightarrow c(L)$$

$R = R, \quad L = L_e$

$$c(L_e) + c(D_{s+j}) \longrightarrow c(L_e)$$

SBU D N

$R = \overline{R}$

$c(U) - c(D) \longrightarrow c(U)$

$R = R$

$c(U) - (D_{s+j}) \longrightarrow c(U)$

SBL D N

$R = \overline{R}, \quad L = L$

$c(L) - c(D) \longrightarrow c(L)$

$R = \overline{R}, \quad L = L_e$

$c(L_e) - c(D_s) \longrightarrow c(L_e)$

$R = R, \quad L = L$

$c(L) - c(D_{s+j}) \longrightarrow c(L)$

$R = R, \quad L = L_e$

$c(L_e) - c(D_{s+j}) \longrightarrow c(L_e)$

RPC-4010 CONSOLE

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
POWER ON	X		A	NO	———	RPC-4010	———	POWER OFF	M
POWER OFF	X			NO	———	RPC-4010	———	POWER ON	M
START COMPUTE	X		N	NO	ONE OPERATION - - - - - START COMPUTE (RPR)	RPC-4010	BOTH COMPUTE & STOP	ONE OPERATION	M
ONE OPERATION	X		A	NO	START COMPUTE - - - - SET INPUT - - - - - EXECUTE LOWER - - - - - START COMPUTE (RPR)	RPC-4430	———	cannot be overridden but see the description page	T

M = momentary T = two-position

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
SET INPUT	X		A	NO	ONE OPERATION EXECUTE LOWER contingent on I/O settings	RPC-4010	———	any start signal terminates effect of SET INPUT and it can only be activated while ONE OPERATION is depressed.	M
EXECUTE LOWER ACCUMULATOR	X		A	NO	ONE OPERATION normally used with SET INPUT	RPC-4010	———	ONE OPERATION must be depressed	T
BRANCH CONTROL	X		only when branch control is on	TBC CXE CMG CME SNS OVERFLOW ONE OPERATION		RPC-4010	is indicator when lighted	if Branch Control is on, depression of this button turns it off in normal mode	M

M = momentary T = two-position

BUTTON OR INDICATOR	USED ON LINE	USED OFF LINE	LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
SENSE SWITCHES	X		A	SNS		RPC-4010	BRANCH CONTROL	———	T
OSCILLOSCOPE REGISTERS	X		——— The oscilloscope provides a visual display of the binary contents of the four machine registers.						
L DISPLAY	X		———	EXC (16 & 32)	———	RPC-4010	OSCILLO-SCOPE	———	MULTI-PLE POSI-TION
STOP	X		when computer is stopped on HLT, INP, or ONE OPERATION	INP HLT	ONE OPERATION - - - - - START COMPUTE	RPC-4010	OPPOSITE OF COMPUTE	ANY START SIGNAL	———

M = momentary T = two-position

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
COMPUTE	X		when computer is executing instructions	HLT STOP CODE OR INP	START COMPUTE (CC) - - - - START COMPUTE (RPR) - - - - ONE OPERATION	RPC-4010	OPPOSITE OF STOP	ONE OPERATION - - - - - HLT - - - - - INP	——

RPC-4430 RIGHT CONTROL PANEL

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
SYSTEM POWER	X	X	when power is on	NO	POWER ON VARIOUS I/O DEVICES IN SYSTEM	ALL I/O DEVICES	VARIOUS	—	T
SINGLE CHARACTER MODE	X		N	INP	ON-LINE INPUT SELECTION	RPC-4430 RPC-4480	CHARACTER LIGHTS - - - - - PARITY MONITOR	NO	T
PARITY MONITOR INHIBIT	X		when bad parity is en-countered	INP	PARITY MONITOR INHIBIT	RPC-4430	CHARACTER LIGHTS - - - - - PARITY MONITOR		
PARITY MONITOR INHIBIT	X		A	INP	PARITY MONITOR INHIBIT	RPC-4430	CHARACTER LIGHTS - - - - - PARITY MONITOR	NO	T

M = momentary T = two-position

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
MASTER RESET	X		N	PRD	ALL ON-LINE I/O SELECTION	RPC-4430 RPC-4480	VARIOUS	SUBSEQUENT ON-LINE SELECTION	M
INPUT DUPLICATION SELECT (COPY MODE)	X		A	PRD	INPUT DUPLICATION RESET	RPC-4430 RPC-4480	VARIOUS	SUBSEQUENT INPUT DUPLICATION RESET	M
INPUT DUPLICATION RESET (COPY MODE)	X			PRD	INPUT DUPLICATION SELECT	RPC-4430 RPC-4480	VARIOUS	SUBSEQUENT INPUT DUPLICATION SETTING	M
START READ	X		N	INP PRD	SET INPUT - - - - - ON-LINE INPUT SELECTION	RPC-4430 RPC-4480	SELECTION MONITOR - - - - - PARITY MONITOR & VARIOUS I/O INDICATORS	STOP AHEAD - - - - - - MASTER RESET - - - - - - any start signal	M

M = momentary T = two-position

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
STOP READ	X		N	INP PRD	START READ - - - - - - MASTER RESET - - - - - - INPUT DUPLICATION RESET	RPC-4430 RPC-4480	SELECTION MONITOR - - - - - PARITY MONITOR	START READ INP	M
START COMPUTE	X		N		START COMPUTE (CC)	RPC-4010	STOP (CC) - - - - COMPUTE (CC)		M
CHARACTER INDICATOR LIGHTS	X		when the on-line reader detects punch in corre-sponding channel	INP	PARITY MONITOR INHIBIT & RESET	RPC-4430	PARITY MONITOR	PARITY MONITOR RESET & INHIBIT	—

M = momentary T = two-position

RPC-4430 READER PUNCH LEFT CONTROL PANEL

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
POWER ON	X	X			SYSTEM POWER	RPC-4430 RPC-4480		SYSTEM POWER	T
SELECTION MONITOR	X	X	when the same device(s) is selected on & off line	NO	ALL ON-LINE & OFF-LINE SELECTIONS	RPC-4010 RPC-4430 RPC-4480	VARIOUS SELECTION INDICATORS	CORRECTIVE ACTION REQUIRED	
TYPEWRITER TO COMPUTER	X		A	PRD INP	START READ - - - - - - STOP READ - - - - - - OFF-LINE TYPEWRITER SELECT - - - - - - READER TO COMPUTER	RPC-4010 RPC-4430 RPC-4480	SELECTION MONITOR	MASTER RESET OFF-LINE TYPEWRITER SELECT PRD SUBSEQUENT READER TO COMPUTER	M

M = momentary T = two-position

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
READER TO COMPUTER	X		A	PRD INP	START READ - - - - - - STOP READ - - - - - - OFF-LINE READER SELECT TYPEWRITER TO COMPUTER	RPC-4010 RPC-4430 RPC-4480	SELECTION MONITOR	MASTER RESET - - - - - - OFF-LINE TYPEWRITER SELECT - - - - - - SUBSEQUENT TYPEWRITER TO COMPUTER - - - - - - PRD	M
AUX. TYPEWRITER TO COMPUTER	X		A	PRD INP	START READ - - - - - - STOP READ	RPC-4010 RPC-4430 RPC-4480	SELECTION MONITOR	MASTER RESET - - - - - SUBSEQUENT READER TO COMPUTER - - - - - PRD	M

M = momentary t = two-position

BUTTON OR INDICATOR	USED ON LINE	USED OFF LINE	LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
COMPUTER TO TYPEWRITER	X		A	PRD PRU	VARIOUS ON-LINE SELECTION - - - - - OFF-LINE TYPEWRITER SELECT - - - - - COPY MODE	RPC-4010 RPC-4430 RPC-4480	SELECTION MONITOR	MASTER RESET - - - - - - OFF-LINE TYPEWRITER SELECT - - - - - - PRD	M
COMPUTER TO PUNCH	X		A	PRD PRU	VARIOUS ON-LINE SELECTION - - - - - - COPY MODE - - - - - - OFF-LINE PUNCH SELECT	RPC-4010 RPC-4430 RPC-4480	SELECTION MONITOR	PRD - - - - - - MASTER RESET - - - - - - OFF-LINE PUNCH SELECT	M
COMPUTER TO AUX. TYPEWRITER	X		A	PRD PRU	VARIOUS ON-LINE SELECTION - - - - - - COPY MODE	RPC-4010 RPC-4430 RPC-4480	SELECTION MONITOR	MASTER RESET - - - - - PRD	M

M = momentary t = two-position

BUTTON OR INDICATOR	USED ON LINE	USED OFF LINE	LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
READER TAPE MONITOR	X	X	when reader is out of tape or tape is taut	PRD		RPC-4430 RPC-4010			
PUNCH TAPE MONITOR	X	X	WHEN TAPE IS TAUT	SNS 64 PRD	ON-OFF-LINE PUNCH SELECT - - - - - - TAPE FEED	RPC-4430 RPC-4480			
CONDITIONAL STOP		X	A		OFF-LINE START READ - - - - - - (RPL) VARIOUS OFF-LINE INPUT SELECT - - - - - - SINGLE CHARACTER MODE (RPL)	RPC-4430 RPC-4480 OFF-LINE INPUT DEVICES			T

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
TAPE FEED		X	N		OFF-LINE PUNCH SELECT MUST BE ACTIVE	RPC-4430	PUNCH TAPE MONITOR	DESELECTION OF PUNCH	M
TYPEWRITER SELECT	X	X	A	PRD INP	TYPEWRITER TO COMPUTER - - - - - - COMPUTER TO TYPEWRITER - - - - - - OFF-LINE READER SELECT	RPC-4430 RPC-4010 RPC-4480	SELECTION MONITOR	if the reader is selected off-line the typewriter (if selected) can only be used for off-line output.	
READER SELECT	X	X	A	PRD INP	READER TO COMPUTER - - - - - - OFF-LINE TYPEWRITER SELECT	RPC-4430 RPC-4480 RPC-4010	SELECTION MONITOR - - - - - READER TAPE MONITOR	NO	T

M = momentary T = two-position

BUTTON OR INDICATOR	USED ON LINE	USED OFF LINE	LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
PUNCH SELECT	X	X	A	PRD PRU	COMPUTER TO PUNCH	RPC-4430 RPC-4010	SELECTION MONITOR - - - - - PUNCH TAPE MONITOR	NO	T
START READ		X	N	NO	SINGLE CHARACTER MODE (RPL) - - - - - - CONDITIONAL STOP (RPL) - - - - - - OFF-LINE INPUT SELECTION	RPC-4430 RPC-4480	READER TAPE MONITOR	STOP READ (RPL)	M
STOP READ		X	N	NO	OFF-LINE INPUT SELECTION	RPC-4430 RPC-4480	NO	START READ (RPL)	M

M = momentary T = two-position

BUTTON OR INDICATOR	USED		LIGHTED ACTIVE OR NORMAL	ASSOCIATED WITH INSTRUCTION	ASSOCIATED WITH BUTTON	AFFECTS UNIT	ASSOCIATED WITH INDICATOR	OVERRIDDEN BY BUTTON OR INSTRUCTION	TYPE OF SWITCH
	ON LINE	OFF LINE							
SINGLE CHARACTER MODE		X	A	NO	OFF-LINE INPUT SELECTION - - - - - START READ	RPC-4430 RPC-4480	NO	CONDITIONAL STOP (RPL)	T

M = momentary T = two-position

RPC-4000 INPUT/OUTPUT SELECTION CODES FOR INSTRUCTION PRD

D track	Input Selected	Output
64	Reader	
65	Reader	Punch
66	Reader	Typewriter
67	Reader	Punch & Typewriter
68	Typewriter	
69	Typewriter	Punch
70	Typewriter	Typewriter
71	Typewriter	Punch & Typewriter
72	Photo--Fwd & Search	
73	Photo--Rev & Search	
74	Photo--Fwd	
75	Photo-Rev	
76-94	Available for additional units--probably input	
95	Master Reset--Reset all units	
96	Available	
97		Punch
98		Typewriter
99		Punch & Typewriter
100		
101		Punch
102		Typewriter
103		Punch & Typewriter
104, 105	Search mode	
106		High Speed Punch
107-124	Available, probably for output units	
125	Copy mode on	
126	Copy mode off	
127	Reset output units	

NOTES:

1. Selection of a new input device automatically resets the previous one. Only one input device may be in the system at a time.
2. Output devices may be added in any combination that they are included in the user's possession. A reset command is necessary to drop an output device from the system.
3. Because the basic system is standard, multiple selection codes have been included for sections of it.

ALPHANUMERIC CODES

The following list gives the tape codes and the computer's internal configurations of the typewriter keyboard.

NUMERIC	DEFINITION	BINARY	NUMERIC	DEFINITION	BINARY
00	Tape feed	000000	32	g G	100000
01	Carriage return	000001	33	h H	100001
02	Tab	000010	34	i I	100010
03	Backspace	000011	35	j J	100011
04		000100	36	k K	100100
05	Upper Case	000101	37	l L	100101
06	Lower Case	000110	38	m M	100110
07	Line feed	000111	39	n N	100111
08	*Stop Code	001000	40	o O	101000
09		001001	41	p P	101001
10		001010	42	q Q	101010
11	Photo Reader	001011	43	r R	101011
12		001100	44	s S	101100
13	End of Block	001101	45	t T	101101
14		001110	46	u U	101110
15		001111	47	v V	101111
16	0)	010000	48	w W	110000
17	1 o	010001	49	x X	110001
18	2 "	010010	50	y Y	110010
19	3 #	010011	51	z Z	110011
20	4	010100	52	, $	110100
21	5 △	010101	53	= :	110101
22	6 @	010110	54	[;	110110
23	7 &	010111	55	] %	110111
24	8 '	011000	56		111000
25	9 (	011001	57		111001
26	a A	011010	58	+ ?	111010
27	b B	011011	59	-	111011
28	c C	011100	60	. —	111100
29	d D	011101	61	Space	111101
30	e E	011110	62	/ ÷	111110
31	f F	011111	63	Code delete	111111

TABLE OF BASIC EXC DATA-TRACK SETTINGS

COMMAND	DATA ADDRESS	NEXT ADDRESS	EFFECT
EXC	098	ANY	No Operation
EXC	198	ANY	Exchange UPPER into LOWER
EXC	298	ANY	Exchange LOWER into UPPER
EXC	498	ANY	Exchange UPPER into INDEX. If this instruction is repeated it is an effective No Operation.
EXC	898	ANY	Exchange INDEX into UPPER
EXC	1698	ANY	Change LOWER to 8-words
EXC	3298	ANY	Change LOWER to 1 word
EXC	4898	ANY	Change state of LOWER
EXC	6498	ANY	RESERVED, if used it is at present an effective No Operation

MODULO 8 TABLE

RECRC0 or RECRC8	00	08	16	24	32	40	48	56
RECRC1	01	09	17	25	33	41	49	57
RECRC2	02	10	18	26	34	42	50	58
RECRC3	03	11	19	27	35	43	51	59
RECRC4	04	12	20	28	36	44	52	60
RECRC5	05	13	21	29	37	45	53	61
RECRC6	06	14	22	30	38	46	54	62
RECRC7	07	15	23	31	39	47	55	63

TABLE OF POWERS OF 2

AND POWERS OF 16

	2^N	N	2^{-N}	
16^0	1	0	1.0	16^{-0}
	2	1	0.5	
	4	2	0.25	
	8	3	0.125	
16^1	16	4	0.062 5	16^{-1}
	32	5	0.031 25	
	64	6	0.015 625	
	128	7	0.007 812 5	
16^2	256	8	0.003 906 25	16^{-2}
	512	9	0.001 953 125	
	1 024	10	0.000 976 562 5	
	2 048	11	0.000 488 281 25	
16^3	4 096	12	0.000 244 140 625	16^{-3}
	8 192	13	0.000 122 070 312 5	
	16 384	14	0.000 061 035 156 25	
	32 768	15	0.000 030 517 578 125	
16^4	65 536	16	0.000 015 258 789 062 5	16^{-4}
	131 072	17	0.000 007 629 394 531 25	
	262 144	18	0.000 003 814 697 264 625	
	524 288	19	0.000 001 907 348 632 812 5	
16^5	1 048 576	20	0.000 000 953 674 316 406 25	16^{-5}
	2 097 152	21	0.000 000 476 837 158 203 125	
	4 194 304	22	0.000 000 238 418 579 101 562 5	
	8 388 608	23	0.000 000 119 209 289 550 781 25	
16^6	16 777 216	24	0.000 000 059 604 644 775 390 625	16^{-6}
	33 554 432	25	0.000 000 029 802 322 387 695 312 5	
	67 108 864	26	0.000 000 014 901 161 193 847 656 25	
	134 217 728	27	0.000 000 007 450 580 596 923 828 125	
16^7	268 435 456	28	0.000 000 003 725 290 298 461 914 062 5	16^{-7}
	536 870 912	29	0.000 000 001 862 645 149 230 957 031 25	
	1 073 741 828	30	0.000 000 000 931 322 574 615 478 515 625	
	2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5	