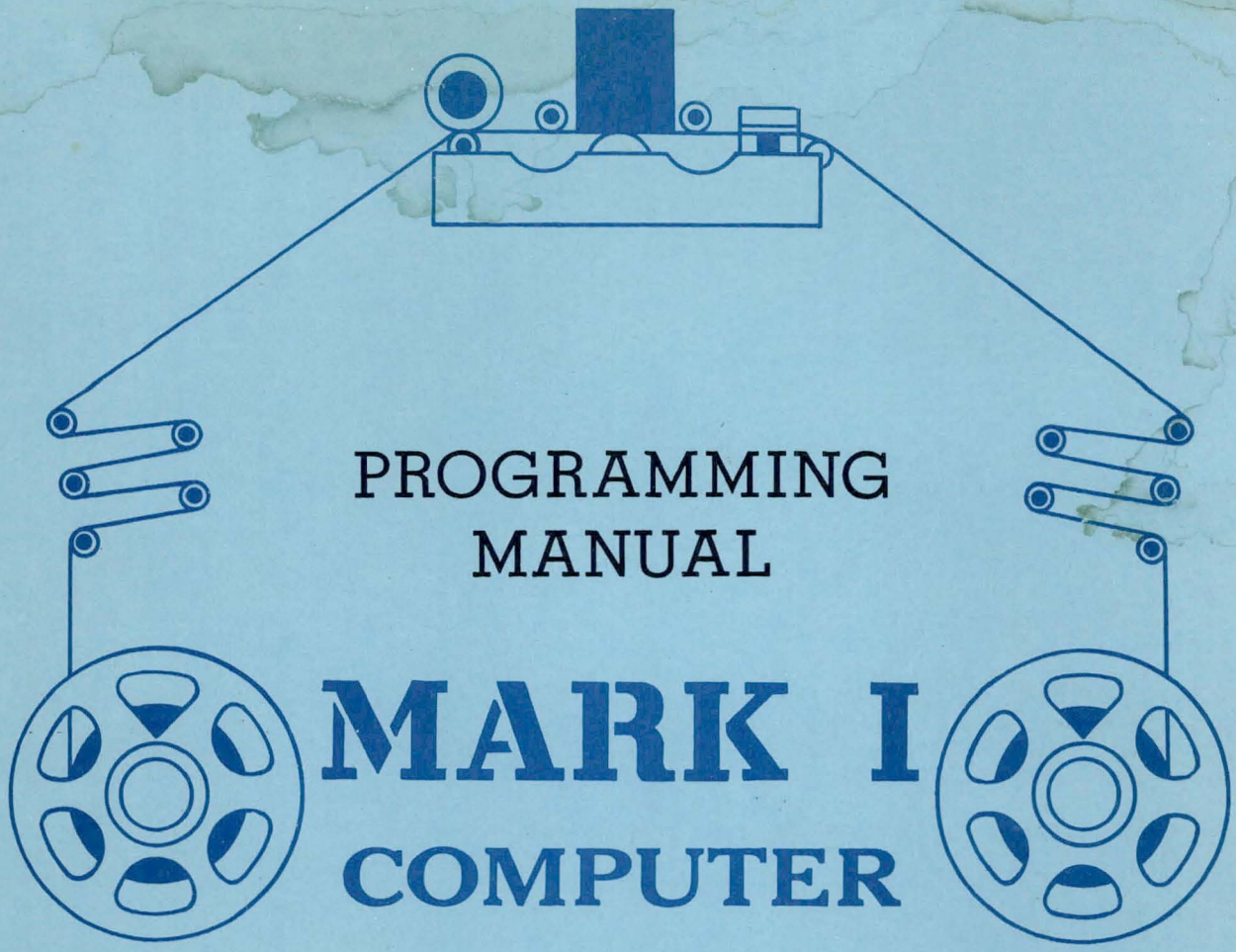


PROGRAMMING MANUAL

MARK I COMPUTER

LINK DIVISION
GENERAL PRECISION INC.
SIMULATION & CONTROL GROUP
BINGHAMTON, NEW YORK

MARK I

PROGRAM MANUAL

TWA

Prepared by

Link Group
General Precision, Inc.
Binghamton, N.Y.

LP1570-5
30 JULY 1964
24-7-64
Printed in U.S.A.

MARK I

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont)

LIST OF ILLUSTRATIONS (Cont)

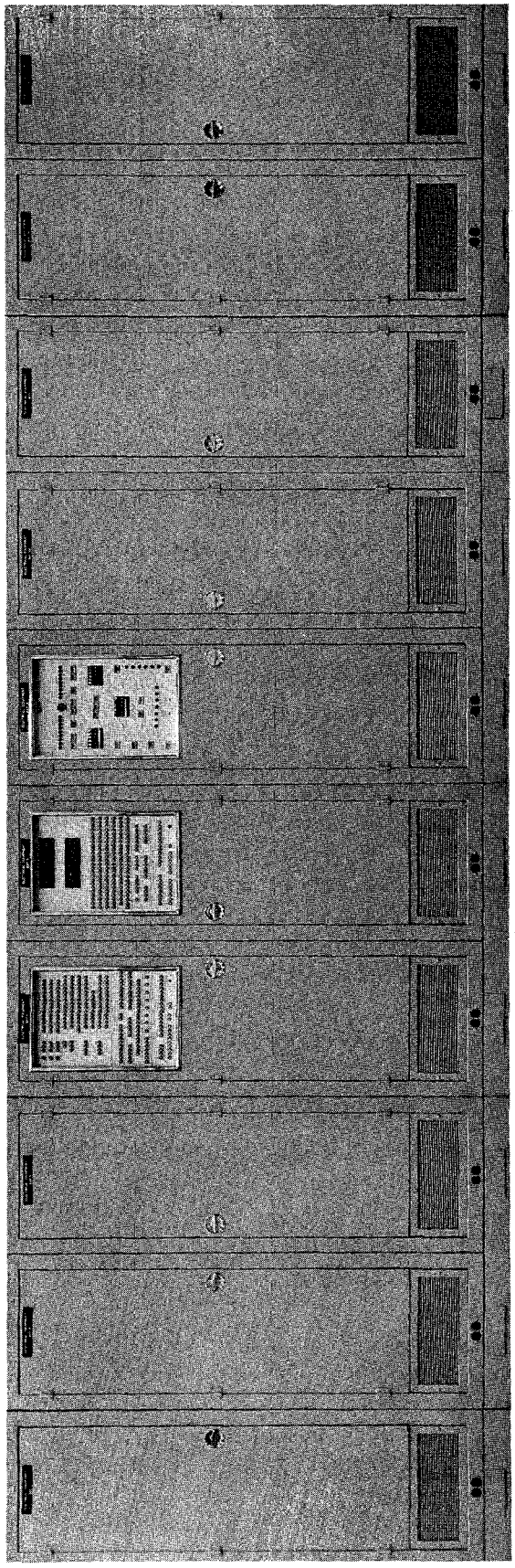


Figure 1-1. Mark I Digital Simulation Computer

SECTION I

COMPUTER CHARACTERISTICS

1-1. GENERAL. The Mark I computer (figure 1-1), operates according to a written program stored on a magnetic drum. The instructions constituting a typical flight simulator program are stored on the drum, using a paper tape reader and drum loader, in the order in which they are to be performed. In this way, during the operation of the computer, these instructions are read and performed in the order in which they were written without waiting and without the necessity of addressing the location of the next instruction.

1-2. If an instruction is read indicating an arithmetic operation, this operation is performed in the main arithmetic unit. This unit contains the registers and logic circuitry for performing all arithmetic operations. Similarly, instructions indicating Boolean operations are performed in the Boolean arithmetic unit. The source of all numerical and Boolean data words for these operations is the main core memory, a fast, random-access storage unit that serves as the working memory of the Mark I.

1-3. The Mark I also includes a 16,384 word linear interpolator that operates in parallel with the main program to generate instantaneous values of the many complex functions encountered in aircraft flight and engine computations. These functions are constantly being calculated, recalculated, and stored in preassigned memory locations in the main core memory for use by the main program. Since this is a parallel operation, there is no time lost in the main program calculating these quantities.

1-4. In the Mark I input system, analog and Boolean inputs from the outside world (i.e., simulator controls) are written into preassigned core memory locations, where they may be used by the main program. The output system reads calculated analog and Boolean data out of core memory to activate simulator equipment. These systems also operate in parallel with the main program, eliminating the need for program time to achieve operations.

1-5. Also operating in parallel with the main program is a radio preselection unit. This unit compares the location of the aircraft and the frequency to which each of its receivers is tuned with the location and frequency of 350 possible radio transmitters. It then selects the best possible transmitter, if any, that each receiver should be picking up, and stores numerical data concerning the selected transmitters in preassigned core memory locations for use by the main program. Since this is a parallel operation, there is again no time taken up in the main program.

1-6. Figure 1-2 is a simplified block diagram of the various processes within the Mark I.

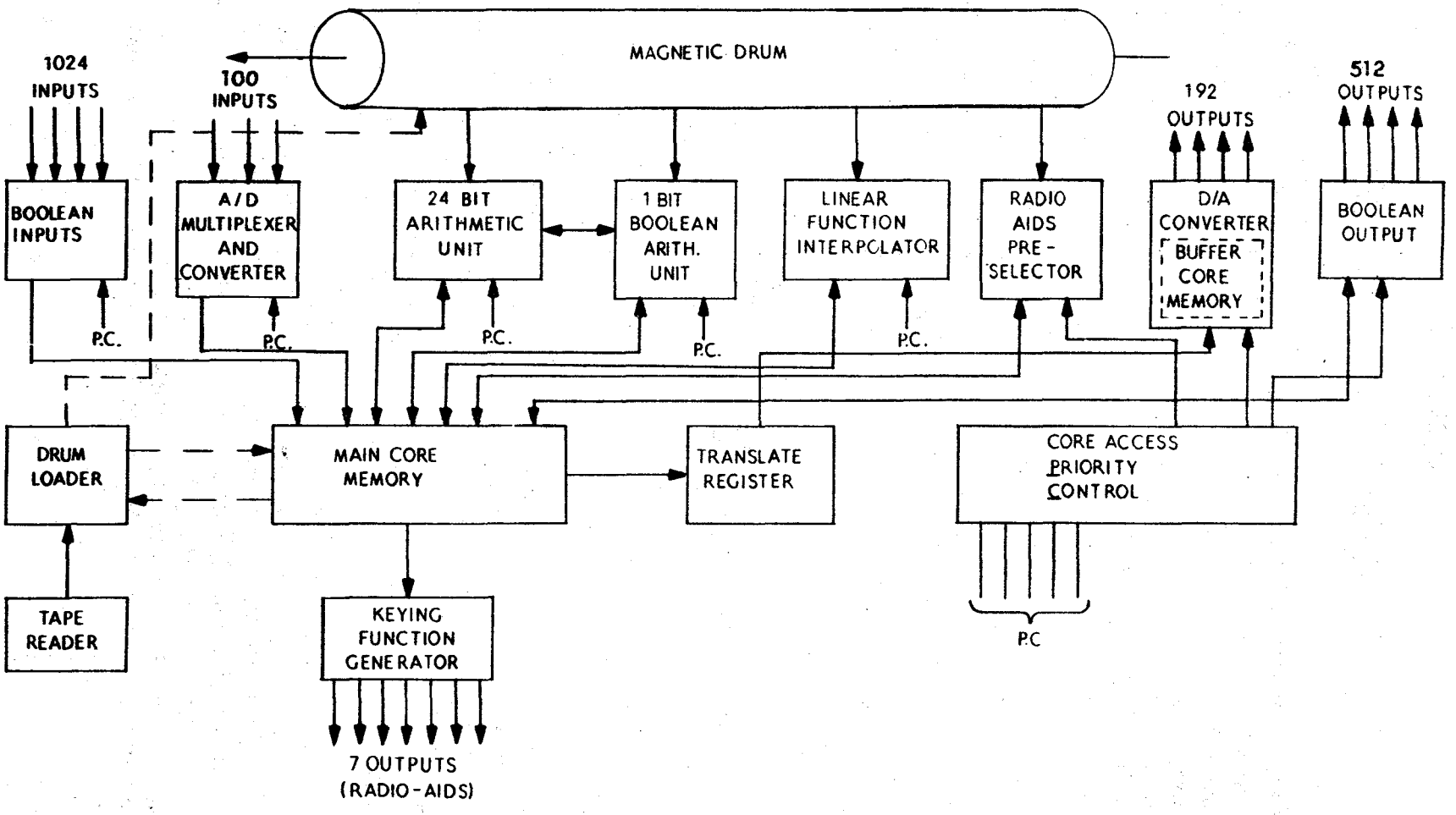


Figure 1-2. Mark I Parallel Processes (Block Diagram)

1-7. MAIN ARITHMETIC UNIT.

1-8. General. The main arithmetic unit acts as the operating center of the computer. This unit consists of a 24-bit accumulator for holding the numerical results of arithmetic operations, all of the necessary logic circuitry for performing arithmetic operations and transferring numerical data, and a salvage register which salvages the old contents of the accumulator when a new word is loaded into it.

1-9. Organization. All numerical operations in the main arithmetic unit are of the fixed-point, binary form. All numbers handled by the main arithmetic unit, either as inputs to or results of arithmetic operations, are in the form of an absolute value and a sign. Sign processes are performed in all arithmetic operations and signs are preserved in the results. All number words are 24 bits in length, the first bit being the algebraic sign and the remaining 23 bits being the absolute value of the binary number.

1-10. If the sign bit is a zero, the number is positive; if it is a one, the number is negative. A typical numerical word in the Mark I is shown in figure 1-3.

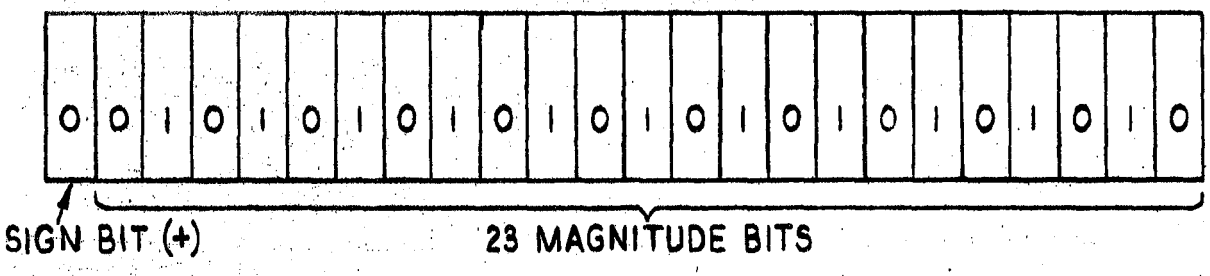


Figure 1-3. Typical 24-Bit Arithmetic Word

1-11. In fixed-point arithmetic, the binary point is assumed to be to the left of the most significant bit (MSB); therefore, the magnitude of the number is always less than 1.0. That is, if the binary point is to the left of the MSB, then the largest magnitude that may be represented is .9999..... in decimal notation or .1111..... in binary notation. All numbers handled by the Mark I must be scaled with the fixed point notation in mind.

1-12. If the result of any arithmetic operation in the Mark I is a number whose magnitude is greater than 1.0, then an automatic overflow process sets each bit of the number equal to one and preserves the sign of the operation. Overflow is possible in the execution of the following instructions: Add, Subtract, Divide, Square Root Step, and Scale Left.

1-13. 24-Bit Arithmetic Accumulator. Almost 60 percent of the instructions for the Mark I cause some operation to be performed on the numerical data contained in the arithmetic accumulator (e.g., add, subtract, multiply, etc.).

1-14. The bit to the left of the binary point in the accumulator is reserved as a sign bit. This bit gives the algebraic sign of the number defined by the remaining 23 bits-zero for a positive number, and one for a negative number.

1-15. Numerical data may be loaded into the accumulator from the core memory and, by virtue of a special instruction, from the drum. Data contained in the accumulator may be stored only in core memory.

1-16. Most operations performed on data in the accumulator may be initiated and finished in one machine cycle (6.105 microseconds). A few of these operations (e.g., multiply or divide) require several machine cycles to complete; therefore, once initiated, care must be taken that no new instructions arrive requiring operations on data in the accumulator until the previous, more lengthy operation has been completed.

1-17. 24-Bit Salvage Register. Associated with the arithmetic accumulator is a salvage register, 24 bits in length. If an instruction is read that directs a data word be loaded from some source into the accumulator, whatever data happens to be in the accumulator at that time would be lost when the new word is loaded. It is the function of the salvage register to store the data word which has been in the accumulator prior to a "LOAD" instruction. Hence, if the number X is in the accumulator at the time an instruction is read directing that Y be loaded into the accumulator, then one machine cycle later Y will appear in the accumulator and X will appear in the salvage register. The previous contents of the salvage register will be lost.

1-18. The salvage register is an addressable location (0000). That is, it may act as a source of data with which to perform arithmetic operations on the contents of the accumulator. However, the salvage register may not be addressed for a "LOAD" instruction.

1-19. MAGNETIC DRUM.

1-20. General. The drum of the Mark I contains sixteen bands of instructions and constants. Once this information has been written onto the drum the "write" equipment may be disconnected and no further information can get into the drum. The design of the Mark I deliberately prevents any further writing on the drum or modification of information contained on the drum.

1-21. Once the computer selects a band of instructions to read, then every word on that band is read in order, at a rate of one word every machine cycle, until all the words on that band have been read. At this time another band is selected and all of those words are read, etc. As each instruction is read, the operation it indicates is performed. If the operation requires more than 6.105 microseconds to complete, it is then necessary to make the immediately following instructions no-operation instructions (NO-OP), until enough time has been allowed to complete the operation.

1-22. The 16 bands on the drum are made up of 240 data tracks, each track being one bit wide and 4096 bits in length around the circumference of the drum. Thus, there is a total storage capacity on the drum of over 983,000 bits. The drum rotates at 2400 RPM and requires 25 milliseconds to complete one revolution.

1-23. All of the bits comprising a word are read in parallel (e.g., if the first word in a band of the main program were being read, then the first bit in each of the 16 data tracks making up that band would be read simultaneously). The read rate of the drum is approximately 164 kilocycles or 6.105 microseconds per word, enabling all words in a band to be read in one drum revolution.

1-24. Of the 16 bands of words written around the drum, (figure 1-4) 11 are for the general program, four contain instructions and constants for the linear interpolator, and one is for radio aids.

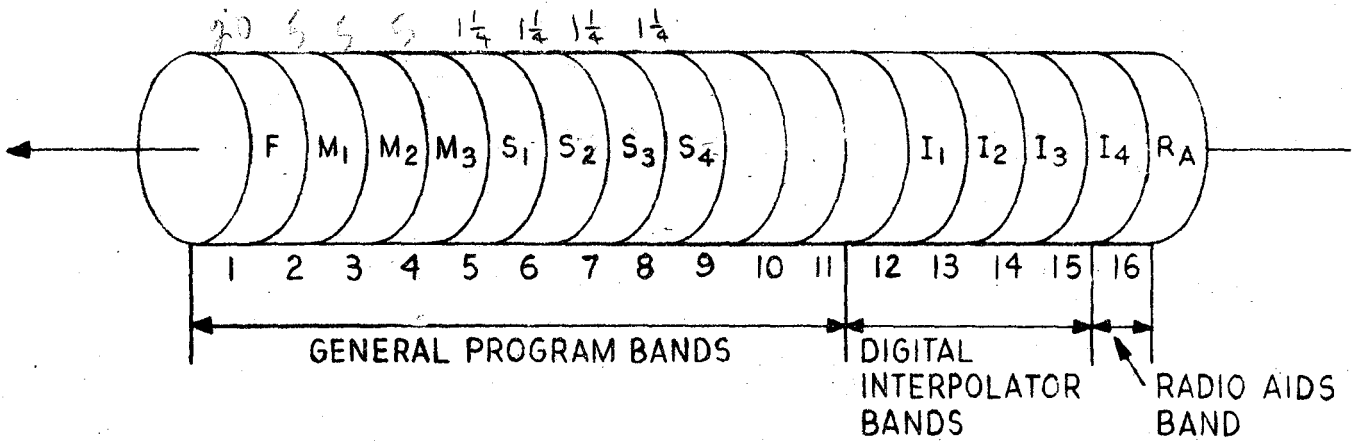


Figure 1-4. Magnetic Drum

1-25. All of the words in the general program portion of the drum are 16 bit words. There are over 45,000 possible words for this portion of the drum (4096 words per band x 11 bands). The four bands of the digital interpolator portion are made up of 11-bit words at 4096 possible words per band. This gives a total of over 16,000 possible words in this section. The 4096 possible words on the radio aids band are all 20-bit words.

1-26. Each time the drum makes one revolution, three bands of instructions and constants are read simultaneously; one band is read and performed by the general program section of the computer, one band by the digital interpolator, and one by the radio aids band.

1-27. The bands comprising the general purpose portion of the drum are not simply read in order from one through 11. The reason for this is that in simulator work some quantities change much more rapidly than others, and consequently, they require a higher frequency response of the computer (i.e., it is essential to recalculate these quantities much more frequently than is necessary for many other computations in the simulation program.

1-28. The Mark I handles these various computations by dividing the general program bands into three categories; fast, medium, and slow bands. Fast band calculations are performed on every other drum rotation (20 times per second). The instructions for the calculation of those quantities which require frequent updating are all placed on this band. Medium fast band instructions will be performed on every eighth revolution of the drum (5 times per second), and slow bands are each read at intervals of 32 drum revolutions (1 time per every 0.8 second).

1-29. Figure 1-5 illustrates the final order in which all of the general program bands are read.

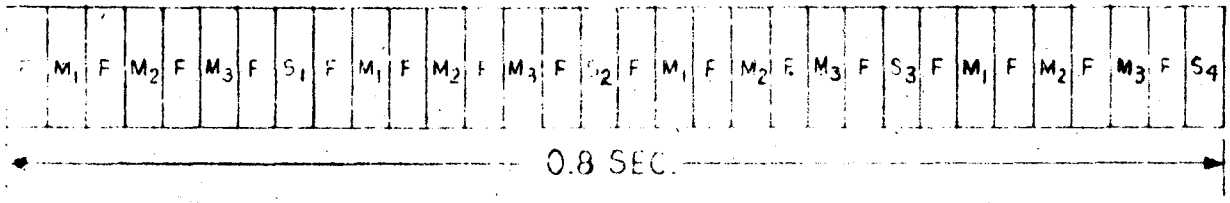


Figure 1-5. General Program Readout Sequence

1-30. Two other bands on the drum are being read simultaneously each time a band of the general program is read (i.e., on each drum revolution). One of these bands is the radio aids band which is read every drum revolution, and the other band is one of the four digital interpolator bands. The four interpolator bands are read in sequence at the rate of one band per drum revolution. Any given interpolator band is read once on every fourth drum revolution.

1-31. Figure 1-6 illustrates the final order in which all of the bands on the drum are read.

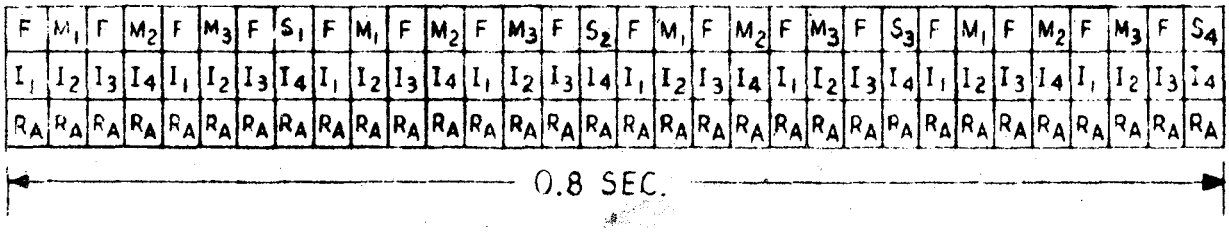


Figure 1-6. Drum Readout

1-32. CORE MEMORY.

1-33. General. The core memory of the Mark I is a 2048 decimal word, random access memory, each of whose 2048 decimal words is an addressable location. Numerical information may be stored in any word location or the information contained in any word location may be read out on command (non-destructive readout). Only one word location in the core may be read out of, or written into during any given machine operate cycle (approximately 6 micro-seconds). Each word of the main core is 24 bits long, of which the most significant bit is the sign bit.

1-34. It is the function of the main core memory to act as the working storage of the computer. That is, all quantities stored in the main core can be changed, updated, erased, etc.

1-35. All the variables in the simulator system equations are each, individually, assigned locations in the core storage, and as each of these variables is recalculated, or changed, the new value is inserted into the proper core location, thus replacing the old outdated value. All inputs to the Mark I computer from the outside world (e.g. cockpit, instructor's station) come to preassigned storage locations in the main core. All outputs from the Mark I to the rest of the simulator system are read out from their preassigned storage locations in the main core. Independent variables which are used by the linear interpolator for the purposes of function generation are read from their assigned location in the core memory. The linear interpolator also stores computed functional values $[f(X), f(X, Y)]$, in preassigned core memory locations. Figure 1-7 shows the core locations for the various quantities stored in the core memory.

1-36. All mathematical quantities that are needed for a simulator program (with the exception of constants which may be stored on the drum), are stored in core memory.

1-37. Priority. Only one word of the entire core memory may be interrogated or written into during any one machine operate cycle. The many parallel processes of the computer (function generation, radio pre-selection, input-output reading, main program arithmetic, etc.) all require memory access. It becomes necessary, therefore that these various processes be given a priority rating. Any operation of the main program requiring memory access takes first priority. Any operations or instructions in the main program that do not require memory access are considered to be "holes" in the main program and it is during these

MAIN CORE

OCTAL WORD ADDRESS	FUNCTION
0000	NOT USED
0001	DIGITAL SWITCH INPUTS (1,024 BOOLEAN WORDS)
0100	
0101	BOOLEAN WORKING MEMORY (496 BOOLEAN WORDS)
0137	
0140	DIGITAL SWITCH OUTPUTS (512 BOOLEAN WORDS)
0177	
0200	A/D INPUTS - 100 WORDS
0375	EXPANDABLE BY 28 WORDS
	0376 & 0377 FOR WORKING MEMORY
0400	D/A OUTPUTS - 128 WORDS
0677	EXPANDABLE BY 64 WORDS
0700	WORKING MEMORY - 64 WORDS
0777	
1000	RADIO AIDS - 80 WORDS
1117	
1120	WORKING MEMORY - 432 WORDS
1777	
2000	WORKING MEMORY &
	CONSTANTS - 1024 WORDS
3777	

INTERPOLATOR ACCESS AVAILABLE

DATA LOST DURING DRUM LOADING

DATA HELD DURING DRUM LOADING

DETAIL OF BOOLEAN CORE

OCTAL WORD ADDRESS MAIN CORE	BOOLEAN CORE	
0000	0-0017	NOT USED
0001	0020	BOOLEAN INPUTS 64 WORDS X 16 BITS PER WORD = 1024 INPUTS
0100	2000 2017	
0101	2020 2037	BOOLEAN WORKING STORAGE 31 X 16 = 496 WORKING STORAGE LOC.
0137	2760 2777	
0140	3000	BOOLEAN OUTPUTS 32 X 16 = 512 OUTPUTS
0177	3777	

NOTE:

1. OCTAL WORDS 3766 - 3777 ARE RESERVED FOR THE ARITHMETIC SCRATCH PAD.
2. BOOLEAN WORDS 2760 2777 ARE RESERVED FOR THE BOOLEAN SCRATCH PAD.

137 - 2760 - 2777

MARK I

Figure 1-7. Core Memory

"holes" that the auxiliary processes of the computer gain access to the core memory. Instructions in the main program such as SCALE, SHIFT, ABSOLUTE VALUE, INVERT, ZERO SLICE, FLAG NEGATIVE, and NO-OPERATION are operations which do not require memory access and thus act as "holes" to auxiliary sections of the computer. Priority for all of the parallel processes of the Mark I is as follows:

a. Main Program

b. Digital Function Interpolation

c. Radio Aids Data Preselector

d. Analog Input Scanning

e. Analog Output Scanning

f. Boolean Output Reading

g. Boolean Input Scanning

1-38. Boolean Storage. Of the 2048 words of memory in the main core, the first 128 words are reserved for use as Boolean storage locations. Only the first 16 bits in each of the 128 words is used for this purpose. This results in a total of 2048 bits of Boolean storage, since a Boolean word is only 1 bit long. The computer is set up to appear as if there were two separate core memory blocks; one being a 2048 word arithmetic core and one being a 2048 word Boolean core. (See figure 1-7.)

1-39. There is a separate Boolean arithmetic unit with its accumulator and salvage register in which all Boolean operations take place. Like its counterpart, the arithmetic core memory, the Boolean core memory acts as the working storage for all Boolean operations. Boolean variables are assigned storage locations here. All Boolean inputs from the rest of the simulator system are read into preassigned storage locations in this memory; and all Boolean outputs from the Mark I to the rest of the simulator system are read out from their assigned storage locations in the Boolean memory.

1-40. Since the Boolean core memory is really made up from 128 words of the main core memory, then any instruction requiring access to the Boolean memory is accessing the main core memory. Any instruction of the main program which requires access, whether it is Arithmetic or Boolean, represents a "highest priority" operation.

1-41. There is no protective circuitry in the Mark I to prevent using a non-Boolean instruction to address the entire contents of a Boolean storage location (the entire

16 bits of one of the 128 main core words). However, if an attempt were made to address the contents of the entire block, it would be meaningless, if the bits had been used for Boolean storage purposes, to try to use it in an ordinary arithmetic operation. It is conceivable, however, that this sort of operation may be deliberately done in the case where it is desired to provide a direct 16-bit binary output of an ordinary arithmetic quantity without going through the Digital to Analog (D/A) converter. In this case, the main word location containing the 16-bits concerned would be "stolen" permanently from the Boolean section for use by the main arithmetic unit. Any attempt now to write Boolean information into any of the 16 bits of the main word location concerned would destroy the meaning of the arithmetic quantity now permanently stored in that main core location. Boolean storage locations may also be stolen (in blocks of 16) for the purpose of storing an externally coded, 16-bit binary word as an input.

1-42. It is important, if such operations as described in paragraph 1-41 are employed for the purpose of providing direct binary inputs and outputs of ordinary arithmetic quantities, that the programmer look on the entire Boolean core as being reduced in size, and must never address any of the bits concerned for any Boolean purpose. (Again, there is no protective circuitry to prevent the programmer from doing so.)

1-43. ANALOG TO DIGITAL CONVERTER.

1-44. General. It is the function of the A/D converter to take all of the analog inputs to the Mark I (from cockpit, etc.) and convert them into 14-bit binary numbers. There are 100 input lines, expandable to 128. Each of these inputs is converted sequentially to binary and stored in its own individual core memory location. This operation is a process which is carried on in parallel with the main program and is fully automatic. There is a block of core memory locations containing the digital equivalent of the various analog input quantities necessary for equations, and all that is required by the programmer is to address the appropriate core location to employ any of these quantities required in computation.

1-45. Scaling. All analog input quantities are scaled in the range of minus 10 volts to plus 10 volts; these reference voltages being provided by the Mark I. The A/D converter converts these numbers to binary numbers scaled from minus one to plus one.

1-46. Priority. The 100 multiplexed inputs of the A/D converter are sampled and converted into properly signed 14-bit binary words in two drum revolutions. The A/D converter will sample inputs only during the first and third quadrant of each drum revolution, and the 100 A/D channels are spread over the four allowed quarters of the two drum revolutions, so that approximately 25 channels are sampled, converted, and stored in each allowed quadrant. A counter provides the core memory addresses in which each quantity is stored.

1-47. Memory access is required for storage, and the A/D converter is under the control of the priority circuitry. (Refer to paragraph 1-37 for A/D priority.) When an input has been sampled, the binary equivalent is converted and held until the next input is sampled. At that time the first input is stored in the address dictated by the counter. The counter is advanced, the next input is held and converted and the process continues until all of the inputs have been sampled. The process repeats endlessly, and occurs without programming attention or instructions.

1-48. DIGITAL TO ANALOG OUTPUT SYSTEM.

1-49. General. Digital to analog conversion in the Mark I operates in parallel with the main program. There are 128 independent analog outputs in the Mark I with the ability to expand to 192 outputs. The analog outputs are used to drive indicators, motion systems, recorders, etc. in the external simulator equipment.

1-50. The equations representing each analog output quantity is implemented in the general program to be computed by the Mark I. A fixed word location in the main core memory is reserved for each quantity, and as each quantity is recalculated in the main program, its new value is stored in its respective memory location. The block of core memory locations which contains the analog output quantities is up-dated twice every drum revolution and is sequentially transferred to a buffer core memory. The words in the buffer core are then fed to individual D/A converters whose outputs appear as analog voltages representing the digital quantities.

1-51. Storage locations in the main core which are reserved for output quantities are all sampled by the buffer core at a rate of 80 times a second. The buffer core, in turn, is sampled by the D/A converter 80 times per second, which represents the sampling rate of the D/A output system.

1-52. In order that a maximum of 192 words in the main core memory be read into the buffer core memory, 192 separate memory accesses are required. All of these words will be accessed once during the second quarter of drum revolution, and again during the fourth quarter of drum revolution. Since each word is read twice during each drum revolution (25 milliseconds), the sampling rate is 80 times per second. The drum plays no actual part in this process, but provides a time base for examining the D/A conversion.

1-53. The time required to read each word is 6.1 microseconds (basic machine cycle), therefore one-quarter revolution of the drum represents enough time for 1024 possible memory accesses. The D/A converter has access to core memory, only when the core memory is not being accessed by either the main program, the digital interpolator or the data preselector. There must be at least 192 "holes" in the total access requirements on the main core memory during the second and fourth quarters of a drum revolution or else all of the output words would not get sampled. A counter in the Mark I generates the core memory

addresses of output quantities which are to be read into the buffer core. As soon as the first address is accessed, the counter advances to the next address and holds that address until an opportunity occurs to interrogate that word. The word is read into the buffer core and the counter advances again. This process continues until all 192 addresses have been interrogated.

1-54. Although the word length in the main core is 23 bits plus one bit for sign, only the sign and the nine most significant bits are read into the buffer core, utilizing only ten of the available 16 bits per word. As words are read out of the main core they are read into a 12 bit translate register. Bits 10, 11 and 12 go through a warbler circuit to produce the round-off bit while bits 1 (2^0), 2 (2^{-1}), and 3 (2^{-2}) are divided by two and drive the vertical wires strung through the three most significant and three least significant bit positions of all 192 words of the buffer core (see figure 1-8). The remaining bits drive the vertical wire, which is strung through the corresponding bit positions of all 192 words of the buffer core.

1-55. BOOLEAN ARITHMETIC UNIT.

1-56. General. The Boolean arithmetic unit is the Boolean counterpart of the main arithmetic unit. It is the purpose of this unit to perform all Boolean operations indicated by instructions in the general program. The logic circuitry of the Boolean arithmetic unit is arranged to perform the functions of "AND" (multiply), "OR" (sum), "INVERT" (compliment), "LOAD", and "STORE". As in the main arithmetic unit (paragraph 1-7), the Boolean arithmetic unit also has an accumulator and a salvage register. Boolean words are one bit in length, and the accumulator is a one bit register which is used to hold the results of all Boolean operations. Boolean words may be loaded into the Boolean accumulator from the main core, and information in the Boolean accumulator may be stored in memory locations in the main core. All Boolean operations indicated by instructions in the general program are performed on the contents of the Boolean accumulator by the contents of the address specified in the instruction.

1-57. Salvage Register. The Boolean salvage register performs the same function as the salvage register of the main arithmetic unit (i.e. to salvage the previous contents of the accumulator when a new word is loaded). Unlike the main arithmetic salvage register, which is only a one-word register, the Boolean salvage register can hold four one-bit words, all of which are addressable. By having a multi-word salvage register, this unit may act as an intermediate, temporary storage unit, thus reducing the core memory access requirements of the Boolean arithmetic unit.

1-58. For the purpose of explaining the operation of the salvage register, consider the sequence of events as a series of several "LOAD" instructions read off the drum (this is a most unlikely series of instructions). Assume the Boolean quantity A is in the accumulator when an instruction directing that the quantity B be loaded is read.

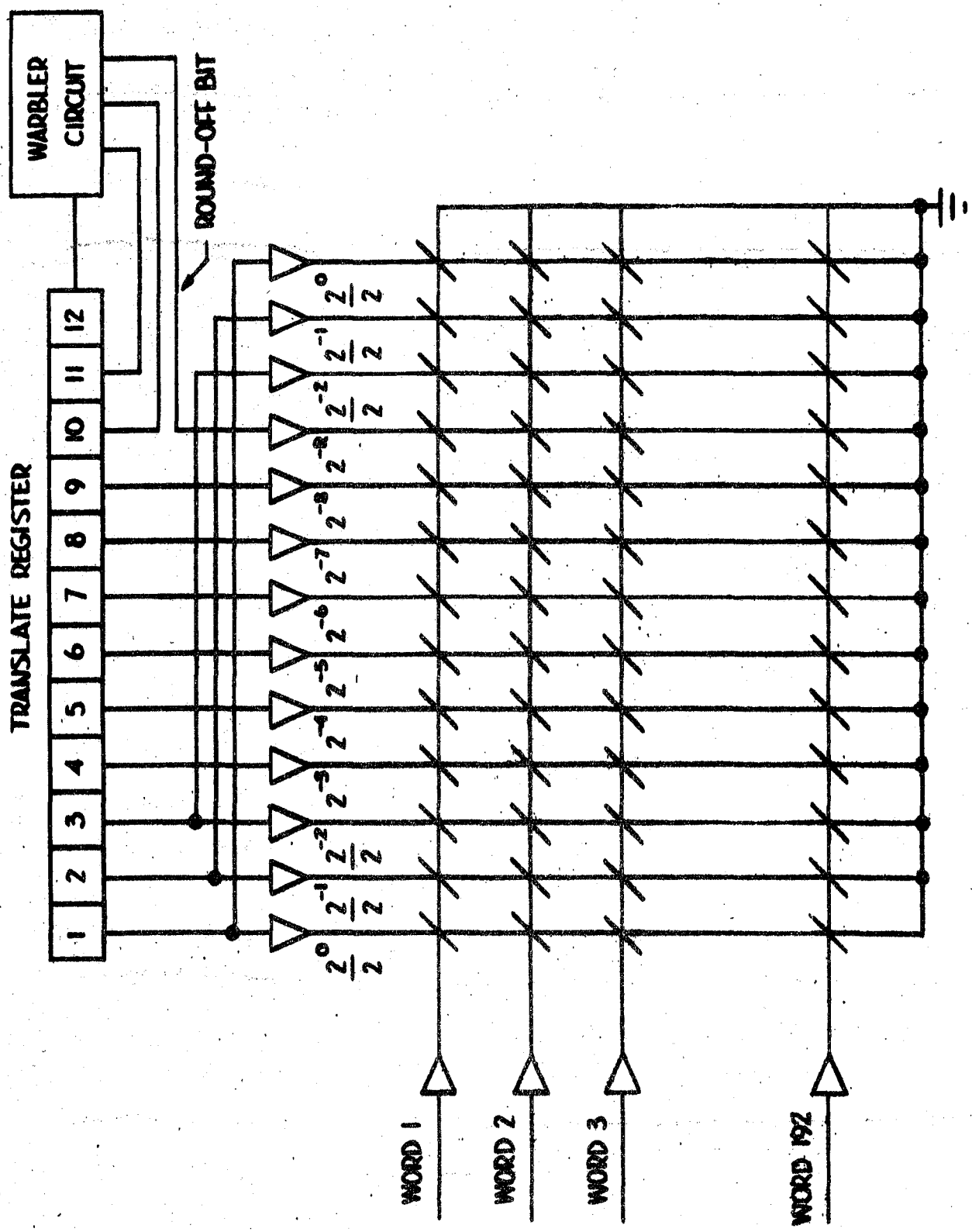


Figure 1-8. 192-Word Buffer Core and Translate Register

Instruction	Contents of Accumulator	Boolean Salvage Register 0000	0001	0002	0003
	A				
Load B	B	A			
Load C	C	B	A		
Load D	D	C	B	A	
Load E	E	D	C	B	A
Load F	F	E	D	C	B

1-59. The four words of the Boolean Salvage Register have addresses 0000 through 0003 and the contents of any of these word locations may be used to perform a logical "AND" or "OR" with the contents of the Boolean accumulator. That is, they are addressable locations. The contents of any of these salvage register locations may be "loaded" into the Boolean accumulator. However, the contents of these locations may not be stored in the Boolean core. Only the contents of the accumulator may be stored in the core memory.

1-60. DIGITAL LINEAR FUNCTION INTERPOLATOR.

1-61. General. Function generation in the Mark I is a process which is done continuously, in parallel with the main program of the machine, and thus, because of the saving of computation time in the main program, contributes largely to the real-time dynamic response of the Mark I. Function generation is accomplished by means of linear interpolation between the ordinates of fixed "breakpoints". Figure 1-9 illustrates an arbitrary function of X whose X axis has been divided into eight equal segments. These eight segments are defined by nine breakpoints: 0, 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, and 1.

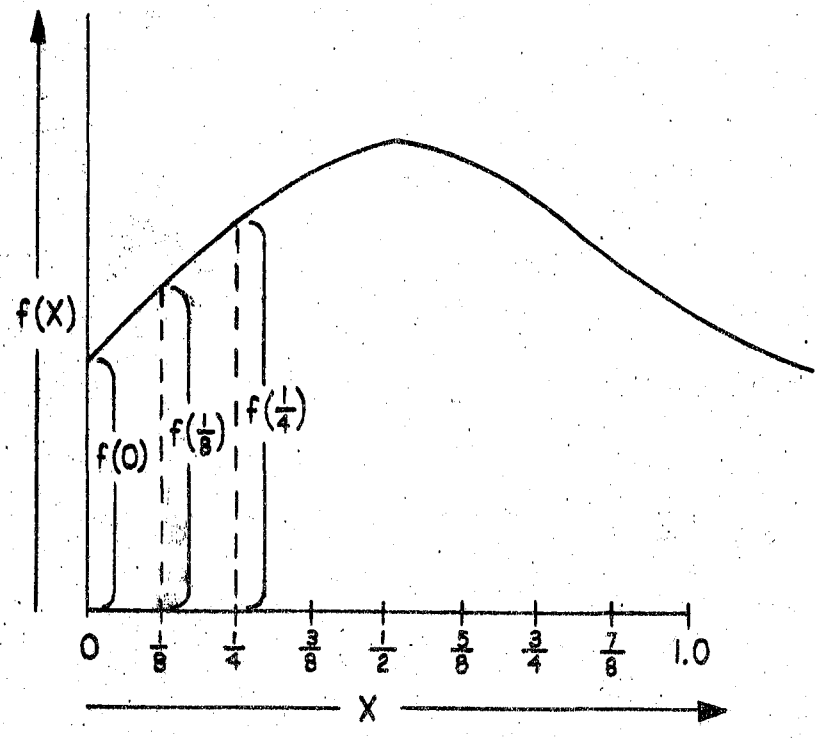


Figure 1-9. Linear Function Curve

1-62. If the ordinates of these nine breakpoints are known, (e.g., f (0), f (1/8), f (1/4), f (3/8), etc.) and if the independent variable X is known, then it is possible to perform a linear interpolation to determine f (X). For instance, if X lies between 1/8 and 1/4 then it is possible to interpolate between f (1/8) and f (1/4) to obtain a very close approximation of f (X).

1-63. The linear interpolator has its own arithmetic unit with a parallel binary adder and the appropriate registers and logic circuitry to do interpolations.

1-64. Scaling. All numbers handled by the Mark I must be scaled so that the magnitude is not greater than one. This also applies to the function interpolator. Therefore, both the independent variable, X, and the ordinate value, f (X), must be scaled so that their magnitudes are not greater than one. All numbers handled by the interpolator are assumed to be positive, so variables and functions must be scaled accordingly.

1-65. Flow Of Information. The four bands on the magnetic drum which are used for function interpolation serve the purpose of storing the ordinates of the breakpoints of all the function curves. The drum also contains the core memory location of the independent variable, and the core memory location in which the calculated value of the function will be stored. Each different function is represented by its own block of information listed on one of the bands, all of the blocks being listed in order around the bands. Figure 1-10 shows the flow of information for function generation.

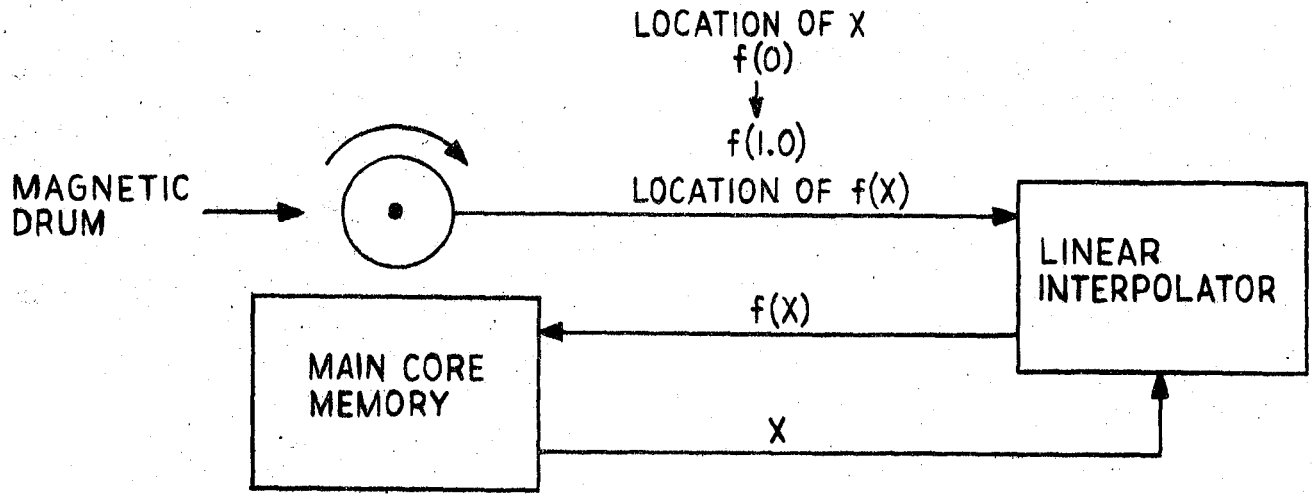


Figure 1-10. Flow of Information (Interpolator)

1-66. It is unnecessary to store the X values of the breakpoints because, since these breakpoints are fixed at 0, 1/8, 1/4, etc., it is only necessary to build the logic of the interpolator in such a way that it can look at the value of X and recognize between which two breakpoints it lies. Consider the binary representation of the numbers, 0, 1/8, 1/4, etc.

```
  0 = 000'0000000
1/8 = 001'0000000
1/4 = 010'0000000
3/8 = 011'0000000
1/2 = 100'0000000
5/8 = 101'0000000
3/4 = 110'0000000
7/8 = 111'0000000
  1 = 111'1111111
```

1-67. An examination of only the first three digits of X will determine between which two breakpoints X lies. If $X < 1/8$ then its first three digits will be 000; hence $0 \leq X < 1/8$. If $1/8 \leq X < 1/4$, then the first three digits will be 001. Figure 1-11 shows how the first three digits of X will determine between which two breakpoints it lies.

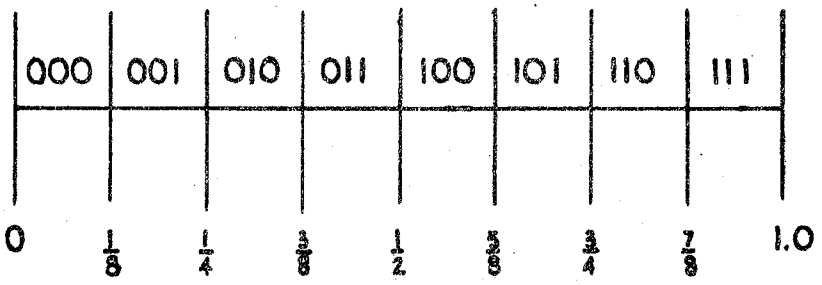


Figure 1-11. Decoding Scheme Determining "Location" of X

1-68. In a block of words on the drum concerning a given function (assume a function of a single variable), the first word that is listed is a control word which serves the purpose of identifying a new function and describing whether it is a function of one, two, or three variables. The second word listed is the memory location of the independent variable. Following this will be the ordinates of the nine breakpoints in order beginning with f (0). The memory location of X, the independent variable, is listed before the ordinate information. Memory access by the main program has priority over access requirements by the interpolator; therefore, the interpolator may have access to memory only during "holes" in the main program. In order to ensure that the current value of the independent variable is obtained from its memory location, its address must be repeated five times. (One of the rules governing the general purpose program is that memory cannot be accessed more than four times in a row or more than 60 percent of the program.)

1-69. Once access to the core memory has been gained, the current value of the independent variable is read into a register in the interpolator. The first three digits of X (the independent variable) are examined to bracket X between two breakpoints. This process is completed before the data field is read off; therefore, before the first ordinate is read, the interpolator already knows which two breakpoints bracket X. As the ordinates are read off the drum, only the two ordinates concerned, f (X_n) and f (X_{n+1}), are held for interpolation. The other ordinate words are ignored by the interpolator. The result of the interpolation is f (X). f (X) is held in the linear interpolator until a word is read off the drum directing that f (X) be stored and giving the location in the core memory in which to store it. The location of f (X) must also be repeated five times to ensure memory access.

1-70. Timing. In the event that X was located between 7/8 and 1.0, the interpolator would have to wait until the eighth and ninth ordinates were read off the drum before it could begin its calculations. This means that some time must be allowed after the last ordinate is listed before the memory location of f (X) is listed. This time is to allow the interpolator to finish its calculations before directing it to store the results.

1-71. In the case of a single variable function, (X), two blank words (12 usec. approximately), allows sufficient time for the interpolator to finish its calculations. In the case of a function of two variables (X, Y), four blank words are necessary, and, for the three variable function (X. Y. Z), six blank words are necessary.

1-72. The final arrangement for the data and instructions for some single variable function (X), is shown in figure 1-12.

1-73. Two Variable Function. A function of two variables as handled by the Mark I will be in the form of a "family" of nine single-variable functions. (See figure 1-13.)

1-74. To describe a two variable function requires 81 ordinates. The address of Y is listed after listing the address of X, and must also be repeated five times to ensure memory access. The 81 data points are listed in zone five beginning with the f (X, Y = 0) curve and ending with the f (X, Y = 1) curve.

1-75. There would be a total of 729 data words for a three variable function: nine data words for X, times nine data words for Y, times nine data words for Z. The address location of the independent variable Z is listed after the Y location and must also be listed five times to ensure memory access.

1-76. Since there are four bands on the drum for the linear interpolator and 4096 words around a single band, there are a total of 16,384 words of storage capacity on the drum to be used for function generation alone. Twenty-two words are required to describe a function of a single variable, so if only single variable

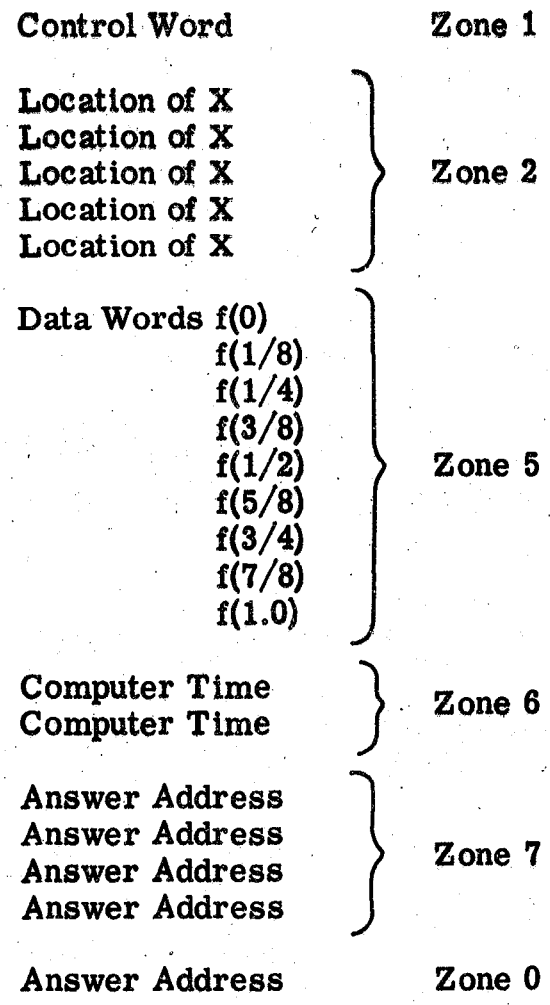


Figure 1-12. Interpolator, Single Variable Function Arrangement

functions were stored on the drum, there would be sufficient capacity to generate over 740 different functions. Since two and three variable functions are also stored, the capacity is affected accordingly.

1-77. It is possible, by using certain indexing bits in the control word, to use the same function data or curve with four different independent variables, the results to be stored in four different locations. This reduces the necessity for having to repeat the data for the same curve several times on the drum, thereby saving storage capacity on the drum. The main "drawback" in using this method is that indexed functions are only recalculated 1/4 as many times as non-indexed functions which are recalculated at a rate of ten times per second.

1-78. All of the words on the four bands of drum storage for the Linear Function Interpolator (LFI) are 11 bits long. Of these 11 bits, one, the MSB, is reserved as a control bit and the remaining ten are for data. Numerical data (ordinates of the

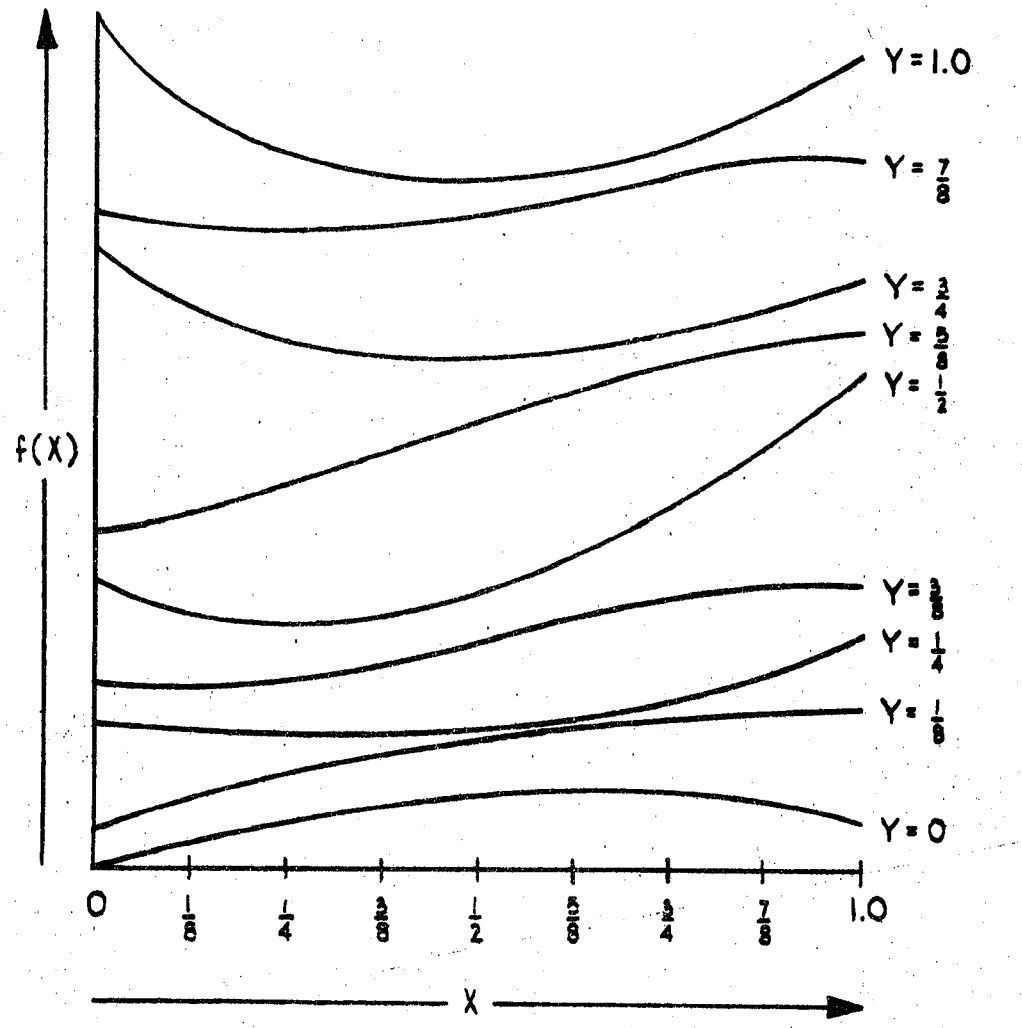


Figure 1-13. Two Variable Function

breakpoints) are only listed with ten binary digit resolution. Arithmetic in the interpolator is carried to 14 places and rounded off.

1-79. RADIO AIDS DATA PRESELECTOR.

1-80. General. It is the function of the Radio Aids Data Preselector to examine a total of 350 different radio transmitters and select the best possible station, if any, that each of the simulated aircraft receivers should be receiving. The Radio Aids Data Preselector operates in parallel with the general program of the Mark I, and it is completely automatic, without need for programmer attention.

1-81. The automatic radio system treats each transmitter as a separate entity. Under this concept, an ILS facility would consist of a localizer transmitter, two 75-megacycle marker transmitters, and two low frequency compass marker transmitters, a total of five separate units.

1-82. Each simulated transmitter is counted separately with the following exceptions:

a. The glideslope facility is provided as a component of the localizer system and does not require a separate transmitter in the total facilities count.

b. A DME (Distance Measuring Equipment) system, if co-located at a VHF Omni-Range Station, is associated (from a computational standpoint) with the related azimuth facility and does not count in the total.

1-83. The 350 available transmitters are divided into five groups according to the type of facility; the maximum number of facilities in each group may not be exceeded, although it is not necessary that all facility channels be employed if a lesser number is desired. The five groups are:

a. Low-frequency transmitters: This group includes low-frequency beacons, low-frequency compass locator facilities and A/N range stations. 127 such facilities can be represented, of which 32 may be A/N range stations.

b. VHF/UHF facilities: These include VOR transmitters, TACAN transmitters, Navy UHF direction finder transmitters and ILS transmitters. 127 independent VHF/UHF facilities can be represented.

c. Outer Markers: The system can represent 32 Outer Markers.

d. Middle Markers: The system can represent 32 Middle Markers.

e. Fan and Z Markers: The system is capable of representing 32 Fan or Z markers which can be intermixed in any desired proportion.

1-84. All information for the 350 transmission facilities is contained in 4096 successive 20-bit parallel words which occur sequentially at 6.105 microsecond intervals around the surface of the drum. Except for the 20-bit length, these words are similar in timing to the instructions and interpolator data contained on the remainder of the Mark I program drum.

1-85. Since all 350 possible facilities must be scanned every drum revolution (40 times per second), preselection is limited to station frequency, latitude coordinate, and longitude coordinate. Preselection in this manner permits the use of simple electronic circuitry and ensures that time-wise, the facility will be inspected.

1-86. Frequency Inspection. For each transmitter represented, an upper and a lower frequency limit is assigned. The facility preselector system will find a particular facility acceptable for a particular receiver from a frequency standpoint, if and only if the frequency of the receiver falls between the upper and lower frequency limits assigned to that facility.

1-87. Although all marker transmitters operate on 75 megacycles, artificial frequencies are assigned in the Mark I. The range of these frequencies are 0 to 1.0, 0 to 0.499, and 0.5 to 1.0. The second two frequency ranges are used to prevent overlaps where more than one marker facility is simulated at any one station.

1-88. Low frequency transmitter facilities are assigned frequency limits by subtracting and adding to the assigned operating frequency a number somewhat larger than half the receiver bandwidth. The resultant two numbers are used as the lower and upper frequency limits.

1-89. VHF/UHF receivers employ digital tuning with discrete frequency assignments. The limits (lower and upper) assigned to the transmitter are numerically close together to ensure that only those transmitters which exactly match the receiver's frequency will pass the frequency test for a given receiver.

1-90. Geographic Position Inspection. The method employed for geographic inspection is assigning two pairs of coordinates to each of the 350 possible facilities. These coordinate pairs represent the upper, lower, left and right boundaries of a rectangle which contains a given facility. These rectangles are arbitrarily assigned so that the left and right boundaries (limits) are in the East-West direction, (longitude axis) and the upper and lower boundaries are in the North-South direction (latitude axis). (See figure 1-14.)

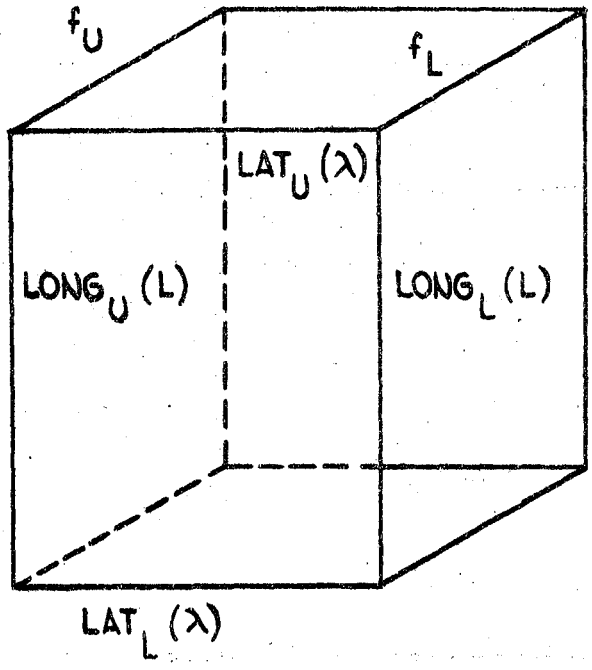


Figure 1-14. Frequency and Geographical Coordinates, Single Transmitter

1-91. Each rectangle is assigned to be as large as possible, although care must be taken to ensure that there are no overlaps of rectangles assigned to different facilities either operating on the same frequency (in the case of markers and VHF/UHF facilities) or operating on adjacent frequencies so that two or more facilities could be within the bandpass of the receiver (in the case of LF facilities).

1-92. The assignment of rectangles to each of the facilities is made by the facility grouping in the computer. For example, there are three groups of markers, Outer, Middle and Fan/Z markers. Even though all three groups operate on the same frequency, only one marker transmitter from each of the three groups can be selected at any one time. Therefore, it is only necessary to assign the rectangles in such a way that for a given group, no overlapping rectangles are assigned. Overlapping rectangles are permitted between facilities of different groups, and because of the programmed calculations involving range and radiation pattern, no interference will result, unless two or more markers of different types that in reality do interfere, are represented and can be received simultaneously.

1-93. Preselection. Preselection of one of the facilities within a group is accomplished by determining whether or not the aircraft is within a pair of left and right Longitude coordinates, a pair of upper and lower Latitude coordinates and whether or not the receiver is tuned to a frequency within a pair of upper and lower frequency limits. Each transmitter is effectively placed within a rectangular box having North/South and East/West boundaries (i.e., the box cannot be placed diagonally on a map). The third dimension of the box is frequency, the three dimensional concept is for convenience is visualizing the physical domain within which the individual transmitters are eligible for reception.

1-94. It is possible to operate in a space free of any of the individual boxes in which case no station may be received. This situation is immediately altered if the receiver tuning is changed since this constitutes an effective movement in the vertical or frequency coordinate in the imaginary three dimensional space, which may result in the preselection of a station whose "box" has been entered.

1-95. When a station of a group meets simultaneously all three preselected criteria, the stored data which completely describes that station is transferred to a specific group of core memory locations associated with the receiver capable of receiving that station. The transferred data is used by the programmer to calculate signal strength, range by orientation, beam pattern, call letters, and so forth, according to the type of facility.

1-96. KEYING FUNCTION GENERATOR.

1-97. General. The Keying Function Generator (KFG) is used to simulate the call letters and characteristic codes of various types of navigational transmitters. The call letters of seven different transmitters can be stored in 8 core memory words. Each word is 16 bits long and four words are utilized for each transmitter, thus providing up to 64 bits per transmitter.

1-98. Priority. The four binary words forming the call letters of each station selected by the Data Preselector are transferred from the drum to core memory. One bit at a time is retrieved from core memory, with a block of seven bits read out during the first seven machine cycles (approximately 42 usec always reserved for KFG priority) of every fourth drum revolution. During this period, one bit is taken from each of the seven stations in storage. The seven bits removed from core memory are shifted into a storage register where they remain for four drum revolutions (0.1 sec). A new set of seven bits is then shifted into the register. These are the next least significant bits (start with the MSB) of each of the seven core memory words being read. This process continues until all four core memory words are read for the seven stations in storage. Since there are 64 bits per station, a total of 6.4 sec is used to obtain all the call letter information stored.

1-99. Since every bit of call letter information represents 0.1 sec of the call letter, a dot is equal to a 1 (0.1 sec), a dash is equal to three 1's (0.3 sec), the space between dots and dashes within a letter is a 0 (0.1 sec), the space between letters within a call letter group is three 0's (0.3 sec), and the space between call groups (if they are repeated in one 64 bit segment) is at least five 0's (0.5 sec).

SECTION II

LIST OF INSTRUCTIONS

2-1. GENERAL. This section contains the 26 machine instructions which the programmer may use in directing the Mark I in a step-by-step solution of an equation. These machine instructions are all that is written on the 11 general-program bands of the drum, and these instructions are read and performed in the order in which they are written. One instruction is read every machine cycle (6.1 microseconds). All instructions describe an operation which the Mark I is to perform, and they give the location of the data word or constant involved in the operation.

2-2. All instructions fall into three general categories: transfer, arithmetic, and control, and each instruction has an identifying code number which is recognized by the machine.

2-3. An instruction word is made up of two sections: a two octal-digit section which gives the operation (identifying code), and a four octal-digit section which specifies a core memory address. Figure 2-1 is an example of an instruction word to add the contents of memory location 1500 to the contents of the accumulator.

01	1500

NOTE: (01 and 1500 are octal numbers)

Figure 2-1. Instruction Word

2-4. When the six octal-digit word (figure 2-1) is punched on the Tape Preparation Unit, it will be automatically converted into a 16-binary-bit word. The first five bits will be the binary code for the operation, and the following 11 bits will be the binary code for the core memory location. (The eleven general-program bands on the drum are all 16 bit wide.) The octal-digit word (figure 2-1) would be coded as shown in figure 2-2.

0	1	1	5	0	0
00	001	01	101	000	000

Figure 2-2. Binary Coded Octal-Digit Instruction Word

2-5. Each digit of the instruction is coded separately. This is the binary-coded octal format. Each octal digit of the instruction is separately converted to binary, and the binary equivalents are written in the same order as the octal numbers.

The first octal digit of the operation section code and address section code never exceeds three and thus requires only two binary digits to simulate it. All of the other octal digits can take on values from 0 to 7 and each requires three binary digits.

2-6. INSTRUCTIONS.

2-7. The following paragraphs contain the complete list of Mark I instructions complete with the instruction code, explanation of the operation, the class in which the instruction falls (control, transfer, or arithmetic), and sample programming procedures for each instruction.

2-8. Load Accumulator.

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	LD	20	MMMM	Transfer the contents of the accumulator to the salvage register. Clear the accumulator. Transfer the contents of core memory address MMMM to the accumulator.

NOTE

The contents of the salvage register may not be loaded in the accumulator. Address 0000 is not allowed.

EXAMPLE: X is in the accumulator and Y is in core memory location 1234 Load Y

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator	Salv.
			X	
LD	20	1234	Y	X

2-9. Store Accumulator.

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	ST	23	MMMM	Transfer the contents of the accumulator to core memory address MMMM. The contents of the accumulator will

Class	Mnemonic Code	O.P. Code	Address	Process
				remain in the accumulator until the next load instruction.

NOTE

Address 0000 is not allowed.

EXAMPLE: Y is in the accumulator. Transfer the contents of the accumulator to core memory address 1000.

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
			Y
ST	23	1000	Y

NOTE

Y is stored in core memory location 1000. Y will also remain in the accumulator until the next LOAD instruction when it will automatically transfer to SALV.

2-10. No-Operation.

Class	Mnemonic Code	O.P. Code	Address	Process
Control	NPA	12	0000	Do not transfer information.
	NPB	25	3777	

NOTE

Indicates an unused instruction word location on the drum. "Time-killer" gives the Mark I time to perform such instructions as multiply, divide, and square root. Priority control is free to accept any auxiliary process. The two NO-OP instructions will be used alternately by the programmer. This instruction also functions as a parity check on the drum read heads. An example of a NO-OP instruction is shown in paragraph 2-13.

2-11. Add.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	ADD	01	XXXX	Transfer the contents of address XXXX to the arithmetic unit and ADD to the contents of the accumulator. The automatic overflow process operates if required.

EXAMPLE: It is desired to form the equation Z = X + Y. Assume Y is stored in memory location 1235, X in memory location 1234, and Z in memory location 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1234	X
ADD	01	1235	X + Y
ST	23	1236	X + Y

NOTE

X + Y is stored in core memory location 1236 (Z). X + Y will also remain in the accumulator until the next LOAD instruction when it will automatically transfer to SALV.

2-12. Subtract.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	SUB	02	XXXX	Transfer the contents of location XXXX to the arithmetic unit and subtact from the contents of the accumulator. Overflow process operates if required.

EXAMPLE: It is desired to form the equation Z = X - Y. Assume X is stored in memory location 1234, Y in memory location 1235, and Z in memory location 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1234	X
SUB	02	1235	X - Y
ST	23	1236	X - Y

NOTE

X - Y is stored in memory location 1236 (Z). X - Y will also remain in the accumulator until the next LOAD instruction when it will automatically transfer to SALV.

2-13. Multiply.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	MLT	03	XXXX	Transfer the contents of location XXXX to the arithmetic unit as multiplicand and multiply with contents of the accumulator as multiplier. The 48-bit product remains in the accumulator and Multiply (MQ) Register. (The 24 most significant bits (MSB's) are in the accumulator.)

NOTE

A multiplication requires an amount of time equal to five machine cycles (30.5 microseconds). Multiplication instructions must be followed by four NO-OP instructions. Boolean instructions may be substituted for the NO-OP instructions since they do not involve the use of the main arithmetic unit.

EXAMPLE: Form the equation W = Z (X + Y). Assume X is in memory location 1234, Y in memory location 1235, Z in memory location 1236, and W in memory location 1237.

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1234	X
ADD	01	1235	X + Y
MLT	03	1236	X + Y
NPA	12	0000	X + Y
NPB	25	3777	X + Y
NPA	12	0000	X + Y
NPB	25	3777	Z (X + Y) = W
ST	23	1237	Z (X + Y) = W

2-14. Negative Multiplication.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	NMT	04	XXXX	Same as multiply except number transferred has sign inverted.

EXAMPLE: Form the equation W = Z - XY. Assume the same memory location for X, Y, Z and W as paragraph 2-13.

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1234	X
NMT	04	1235	X
NPA	12	0000	X
NPB	25	3777	X
NPA	12	0000	X
NPB	25	3777	-XY
ADD	01	1236	Z - XY = W
ST	23	1237	Z - XY = W

2-15. Square.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	SQ	05	0000	Same as multiply except transfer contents of accumulator for use as multiplier and multiplicand.

NOTE

0000 is the only allowed address.

EXAMPLE: Form the equation $Z = X^2 - Y$. Assume X, Y, and Z are in the same memory locations as in paragraph 2-13.

Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1234	X
SQ	05	0000	X
NPA	12	0000	X
NPB	25	3777	X
NPA	12	0000	X
NPB	25	3777	X^2
SUB	02	1235	$X^2 - Y = Z$
ST	23	1236	$X^2 - Y = Z$

2-16. Divide.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	DIV	06	XXXX	Transfer the contents of address XXXX to the main arithmetic unit as divisor and divide into the contents of the accumulator as dividend. Discard remainder of dividend from accumulator and transfer quotient to accumulator. Overflow process operates if required.

NOTE

A divided instruction must be followed by four NO-OP instructions. (Boolean instructions may be substituted for NO-OPS.)

EXAMPLE: Form the equation $W = \frac{XY - Y^2}{Z} + Z$. Assume W, X, Y, and Z are in the same memory location as in paragraph 2-13.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator
LD	20	1234	X
SUB	02	1235	$X - Y$
MLT	03	1235	$X - Y$
NPA	12	0000	$X - Y$
NPB	25	3777	$X - Y$
NPA	12	0000	$X - Y$
NPB	25	3777	$Y (X - Y)$
DIV	06	1236	$Y (X - Y)$
NPA	12	0000	$Y (X - Y)$
NPB	25	3777	$Y (X - Y)$
NPA	12	0000	$Y (X - Y)$
NPB	25	3777	$\frac{Y (X - Y)}{Z}$
ADD	01	1236	$\frac{Y (X - Y)}{Z} + Z =$ $\frac{XY - Y^2}{Z} + Z = W$
ST	23	1237	$\frac{XY - Y^2}{Z} + Z = W$

2-17. Square Root Step.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	SRS	07	MMMM	Take the square root of the contents of the accumulator, by performing one iteration of the Newton-Raphson approximation where $X = \sqrt{Y}$; $X_n = 1/2 \left(\frac{Y}{X_n - 1} + X_n - 1 \right)$ The previous result of this operation, $X_n - 1$, is found in memory location MMMM. The result of this step, X_n, is to be stored in memory location MMMM.

NOTE

The square root of 0 is not allowed. A square root instruction must be followed by four NO-OP instructions (Boolean instructions may be substituted.)

EXAMPLE: Form $Z = X + Y + \sqrt{XY}$. Assume that X is stored in memory location 1233, Y is in 1234 and Z in 1235.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator	Salv.
LD	20	1233	X	
ADD	01	1234	X + Y	
LD	20	1233	X	X + Y
MLT	03	1234	X	X + Y
NPA	12	0000	X	X + Y
NPB	25	3777	X	X + Y
NPA	12	0000	X	X + Y
NPB	25	3777	XY	X + Y
SRS	07	1000	XY	X + Y
NPA	12	0000	XY	X + Y
NPB	25	3777	XY	X + Y
NPA	12	0000	XY	X + Y
NPB	25	3777	$\sqrt{XY}$	X + Y
ST	23	1000	$\sqrt{XY}$	X + Y
ADD	01	0000	$X + Y + \sqrt{XY}$	X + Y
ST	23	1235	Z	

2-18. Scale.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	SCL	10	1---	The address portion of this instruction contains a control code as follows: If the most significant bit (MSB) is 0, left shift. If the MSB is 1, right shift. The remaining portion of the address contains the number of places to be shifted, 001 through 101 (Binary). The MSB of the accumulator will remain unchanged to retain the sign bit.
	SCL	10	0---	

Class	Mnemonic Code	O.P. Code	Address	Process
				The remainder of the accumulator contents will be shifted by the required number of bits, with insertion of zeros starting at the second most significant bit for scale right, and with the insertion of zeros starting at the least significant MQ bit for scale left. For scale left, the MSB of the accumulator (after the sign bit) will be examined before each 1-bit shift. If a "one" exists in this bit, OVERFLOW will occur replacing the scaling process, and ending execution of the instruction.

NOTE

Scaling a number left one place is equivalent to multiplying that number by two. Scaling right one place is equivalent to dividing by two. The Mark I can shift five places in one basic machine cycle (6.1 microseconds).

EXAMPLE: Form $Z = \frac{3X}{2} + 8Y$. Assume X is in memory location 1234, Y in 1235, and Z in 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator	Salv.
LD	20	1234	X	
SCL	10	1001	$\frac{X}{2}$	
ADD	01	1234	$\frac{3X}{2}$	
LD	20	1235	Y	$\frac{3X}{2}$
SCL	10	0003	8Y	$\frac{3X}{2}$
ADD	01	0000	$\frac{3}{2}X + 8Y$	$\frac{3X}{2}$
ST	23	1236	Z	$\frac{3X}{2}$

2-19. Shift.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	SFT SFT	11 11	1--- 0---	Same as the "SCALE" instruction (paragraph 2-18) except that the entire contents of the accumulator, including the sign bit, will be shifted. Zeros are inserted at the LSB for left shifts, and at the MSB for right shifts. The control code is the same as for the scale instruction.

2-20. Invert Sign.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	INS	13	0000	Reverse the sign (the most significant bit) of the accumulator.

NOTE

0000 is the only allowed address.

EXAMPLE: Form Z = X - Y. Assume that Y is already in the accumulator, X is in 1234 and Z is 1235.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator
			Y
SUB	02	1234	Y - X
INS	13	0000	-(Y - X) = X - Y
ST	23	1235	Z

2-21. Absolute Value.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	ABS	14	0000	Make the sign (MSB) of the accumulator positive (0).

NOTE

0000 is the only allowed address.

EXAMPLE: Form Z = /X - Y/. Assume X.2^{-8} is stored in memory location 1234 and Y.2^{-6} is stored in 1235. It is desired to form and store Z.2^{-7} in location 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Scale	Arithmetic Accumulator
LD	20	1235	2^{-6}	Y
SCL	10	1002	2^{-8}	Y
SUB	02	1234	2^{-8}	Y - X
ABS	14	0000	2^{-8}	/Y - X/
SCL	10	0001	2^{-7}	/Y - X/
ST	23	1235	2^{-7}	Z

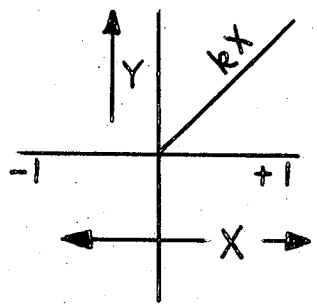
2-22. Zero Slice.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	ZSL	15	0000	If the sign of the number in the accumulator is negative, make the contents of the accumulator zero (including the sign bit). If the sign of the number is positive, do nothing.

NOTE

0000 is the only allowed address.

EXAMPLE: It is desired to form Y = F(X) where /X/ $\leq$ 1 and Y is described by the following curve.



Y = kX for X>0
Y = 0 for X$\leq$0

Assume X located in memory location 1234, k in 1235, and Y in 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator
LD	20	1234	X
MLT	03	1235	X
NPA	12	0000	X
NPB	25	3777	X
NPA	12	0000	X
NPB	25	3777	kX
ZSL	15	0000	
ST	23	1236	Y

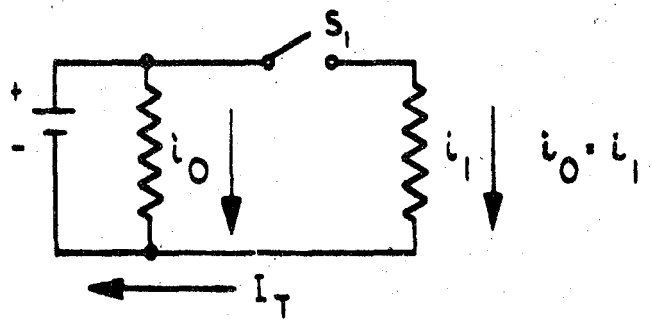
2-23. Conditional Skip.

Class	Mnemonic Code	O.P. Code	Address	Process
Control	SKP	26	The octal equivalent of the number of steps to be skipped.	Skip the next N instructions only if the Boolean accumulator is 1. (Maximum number of skips in one instruction is 127.) The Mark I can skip to another skip instruction.

NOTE

This is the only form of branching in the Mark I.

EXAMPLE: I_T is the total load current on a power buss and the problem is to add i_1, and additional load, to i_0 the minimum load, if switch S_1 is closed. If S_1 is closed $S_1 = 1$. If S_1 is open $S_1 = 0$. Assume S_1 is in the Boolean accumulator, i_0 is in core memory location 1234, and i_1 is in 1235. Calculate I_T and store in 1236.



Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator	Boolean Accumulator
LD	20	1234	i_0	S_1
BIN	32	0000		S_1
SKP	26	0001		S_1
ADD	01	1235	$i_0 + i_1$	S_1
ST	23	1236	$I_T = i_0 + i_1$ or i_0	S_1

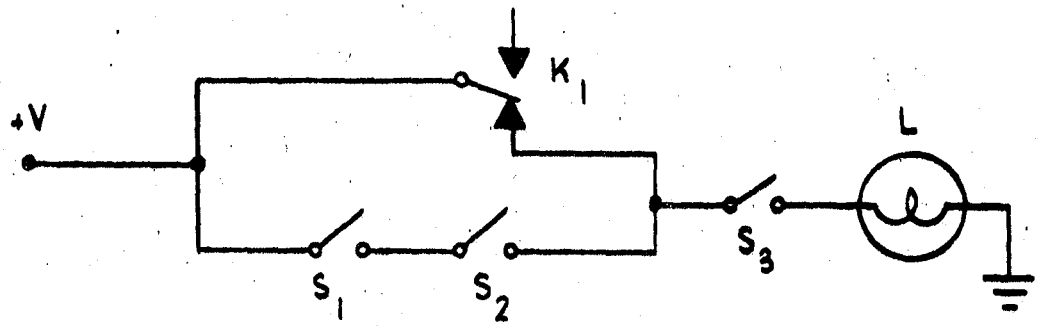
2-24. Invert Boolean Accumulator.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	BIN	32	0000	Complement the contents of the Boolean accumulator (i.e., a 1 is changed to a 0, and a 0 is changed to a 1).

NOTE

0000 is the only allowed address.

EXAMPLE:



$L = S_3 (\bar{K}_1 + S_1 S_2)$

S_1, S_2, S_3, K_1, and L are in storage locations 0100 through 0104.

Mnemonic Code	O.P. Code	Mem. Add.	Boolean Accumulator	Salvage 0000	0001	0002	0003
BLD	33	0103	K_1				
BIN	32	0000	$\bar{K}_1$				
BLD	33	0100	S_1	$\bar{K}_1$			
AND	31	0101	$S_1 S_2$				
OR	30	0000	$\bar{K}_1 + S_1 S_2$	$\bar{K}_1$			
AND	31	0102	$S_3 (K_1 + S_1 S_2)$				
BST	34	0104	L				

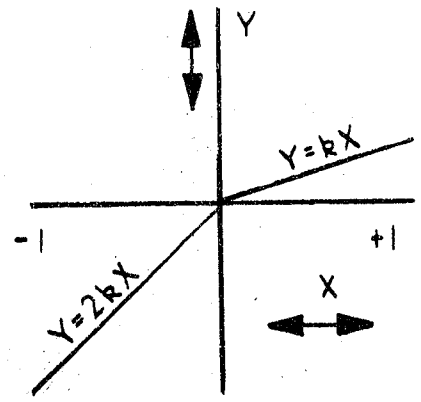
2-25. Flag Negative.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	FLN	16	0000	Set the Boolean accumulator the same as the sign bit in the arithmetic accumulator. The previous contents of the Boolean accumulator are transferred to the salvage register B 0000.

NOTE

0000 is the only allowed address.

EXAMPLE: Generate $Y = kX$ where $0 \leqq X \leqq 1$, and $Y = 2kX$ where $-1 \leqq X < 0$.



Assume X is in location 1234, and k is in 1235. Calculate and store Y in 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator
LD	20	1234	X
MLT	03	1235	X
NPA	12	0000	X
NPB	25	3777	X
NPA	12	0000	X
NPB	25	3777	kX
FLN	16	0000	kX
BIN	32	0000	kX
SKP	26	0001	kX
SCL	10	0001	2kX
ST	23	1236	Y

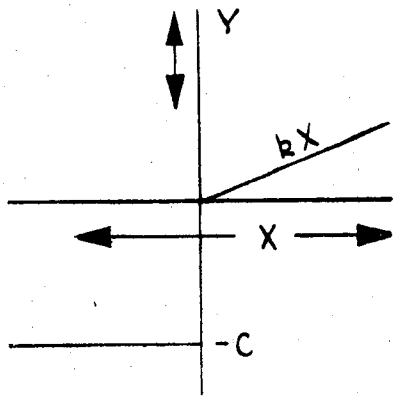
2-26. Index Load.

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	ILD	21	MMMM	Same as LOAD instruction with the addition that the LSB of t e eleven address bits is made to correspond to the conten s of the Boolean accumula or.

NOTE

Address 0000 is not allowed. By use of this instruction, data from either of two locations may be loaded into the accumulator, depending on some external condition (as reflected in the state of the Boolean accumulator). The only pairs of adjacent core memory locations allowed consist of even numbered locations and the next higher number (e.g., 3776 and 3777; 0900 and 0901; etc.).

EXAMPLE: Generate Y = f(X), where f(X) = kX for $0 \leq X \leq 1$ and f(X) = -C for $-1 \leq X < 0$. Assume X is in core memory location 1234, -C/k is in 1235, k is in 1236 and Y in location 1237.



Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1234	X
FLN	16	0000	X
ILD	21	1234	
MLT	03	1236	
NPA	12	0000	
NPB	25	3777	
NPA	12	0000	
NPB	25	3777	f(X)
ST	23	1237	f(X)

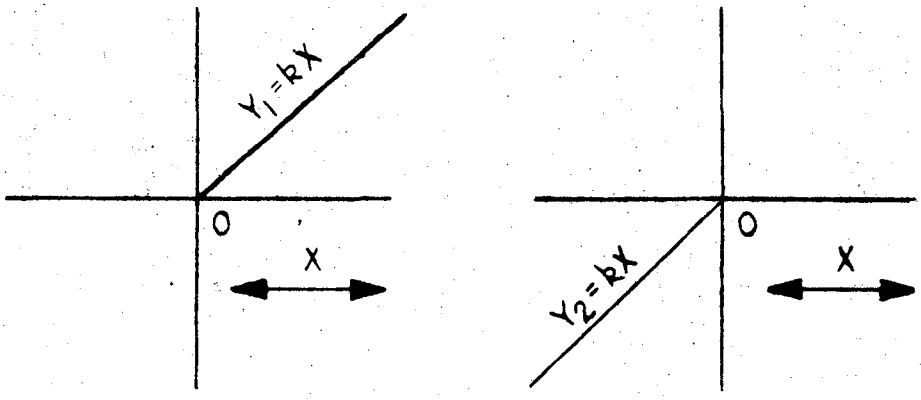
2-27. Index Store.

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	IST	24	MMMM	Same as "STORE" instruction plus the same address modification of "ILD".

NOTE

Address 0000 is not allowed.

EXAMPLE: $Y_1 = kX$ for $X \geq 0$ and $Y_1 = 0$ for $X < 0$
$Y_2 = kX$ for $X < 0$ and $Y_2 = 0$ for $X \geq 0$
Assume X is in location 1230, k in 1231, Y_1 in 1232 and Y_2 in 1233.



Mnemonic Code	O.P. Code	Mem. Add.	Accumulator
LD	20	1230	X
FLN	16	0000	X
MLT	03	1231	
NPA	12	0000	
NPB	25	3777	
NPA	12	0000	
NPB	25	3777	kX
IST	24	1232	

2-28. No Address Load (Load Constant).

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	LDK	22		The address portion of this instruction word contains an 11-bit constant instead of an address. This constant is always assumed to be positive and is loaded into the accumulator. The remaining 12 LSB's of the accumulator are set to zero.

EXAMPLE: Calculate Y = kX + C:
X.2^{-5} is stored in 1234,
Y.2^{-10} is desired.
k.2^{-5} = .86875
C.2^{-3} = .8500

Mnemonic Code	O.P. Code	Mem. Add.	Scale	Accumulator	Salvage
LDK	22	.8687	2^{-5}	k	
MLT	03	1234			
NPA	12	0000			
NPB	25	3777			
NPA	12	0000			
NPB	25	3777	2^{-10}	kX	
LDK	22	.8500	2^{-3}	C	kX.2^{-10}
SCL	10	1005	2^{-8}	C	kX.2^{-10}
SCL	10	1002	2^{-10}	C	
ADD	01	0000	2^{-10}	kX + C	
ST	23	1235	2^{-10}	Y	

2-29. Conditional Stop.

Class	Mnemonic Code	O.P. Code	Address	Process
Control	STP	27	0000	Stop the program. Hold clock counter at the last instruction address. Operates when the "conditional stop" switch on the Mark I is activated.

NOTE

0000 is the only allowed address. This instruction is programmed only in diagnostic routines. It may also be given during any program by use of a manual control on the Mark I. The program may be restarted at the following instruction by means of the conditional stop switch on the Mark I.

2-30. Load Boolean Accumulator.

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	BLD	33	BBBB	Transfer the contents of the Boolean accumulator to Boolean salvage location 0000. Clear the Boolean accumulator and transfer the contents of Boolean memory location BBBB to the Boolean accumulator.

NOTES

1. Boolean addresses, 0000, 0001, 0002, and 0003 are allowed. (These are the addresses of the four Boolean salvage registers.)

2. The address portion of any Boolean instruction word, as written on the drum, is actually a control code which describes the Boolean memory location. This code tells which of the 128 main core memory words (that have been set aside for Boolean storage) is involved, and which of the 16-bits of the word concerned actually is the Boolean word desired. The four LSB's of the 11-bit control code (address portion of the instruction word) are transferred to the Boolean arithmetic unit to provide bit selection on the core memory output register. The first seven bits are transferred to the seven LSB places of the core memory address register and four zeros are inserted into the four most significant places of the register. This process constructs a core memory address from 0000 to 0127 as determined by the MSB's of the control code. The selected word is transferred to the core memory output register where the particular bit concerned is selected by the four LSB's of the control code previously transferred to the Boolean arithmetic accumulator.

2-31. Store Boolean Accumulator.

Class	Mnemonic Code	O.P. Code	Address	Process
Transfer	B ST	34	BBBB	Store the contents of the Boolean accumulator in Boolean core memory location BBBB.

NOTE

Boolean addresses 0000-0003 are not allowed.

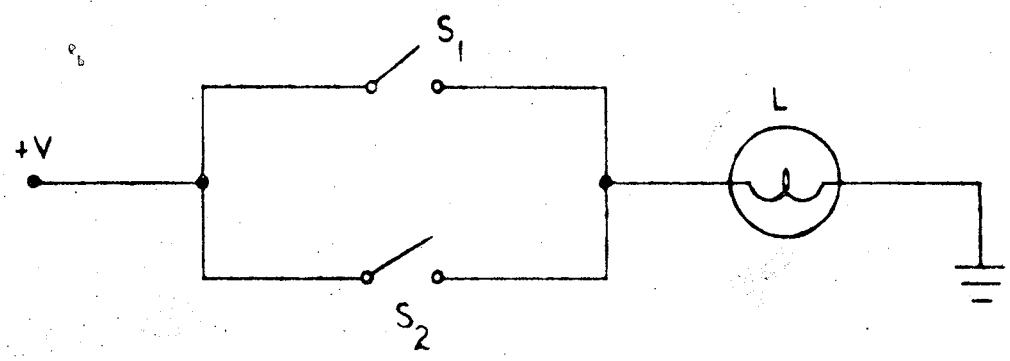
EXAMPLE: X, a quantity which varies from 0 to +1 is stored in 1234. Set a flag to determine whether X is greater or less than 0.5, and store the flag in Boolean core memory location 0100 for later use.

Mnemonic Code	O.P. Code	Mem. Add.	Arithmetic Accumulator	Salvage	Boolean Accumulator
LDK	22	4000	.5000		
LD	20	1234	X	.5000	
SUB	02	0000	X-.5000		
FLN	16	0000	X-.5000		X-.5000 flag
B ST	34	0100	X-.5000		X-.5000 flag

2-32. Boolean Sum.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	OR	30	XXXX	Transfer the addressed bit to the Boolean arithmetic unit and ADD (OR) with the contents of the Boolean accumulator. The sum (OR function) remains in the Boolean accumulator.

EXAMPLE:



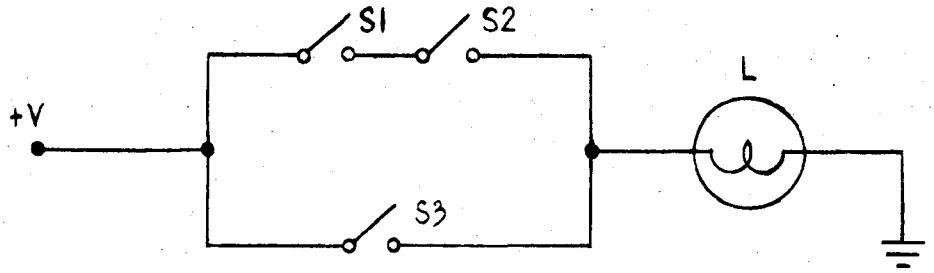
The light, L, is on (1) if S_1 or S_2 are closed (1). $\therefore L = S_1 + S_2$
Calculate L and store. S_1 is in Boolean location 1234, S_2 is in 1235 and L in 1236.

Mnemonic Code	O.P. Code	Mem. Add.	Boolean Accumulator
BLD	33	1234	S_1
OR	30	1235	$S_1 + S_2$
B ST	34	1236	L

2-33. Boolean Product.

Class	Mnemonic Code	O.P. Code	Address	Process
Arithmetic	AND	31	XXXX	Transfer the addressed bit to the Boolean arithmetic unit and multiply (AND) with contents of Boolean accumulator. Products remain in the accumulator.

EXAMPLE:



$L = S_1 S_2 + S_3$ (The light is on (1) when we have either S_1 and S_2 or S_3.)
S_1, S_2, S_3, and L are in Boolean locations 0100-0103

Mnemonic Code	O.P. Code	Mem. Add.	Boolean Accumulator
BLD	33	0100	S_1
AND	31	0101	$S_1 S_2$
OR	30	0102	$S_1 S_2 + S_3$
BST	34	0103	L

2-34. Tape Stop Code.

Class	Mnemonic Code	O.P. Code	Address	Process
External	TRS	37	NONE	A stop code for the photo electric tape reader. Each second character on the tape will be interrogated for this code.

SECTION III

PREPARATION OF PROGRAMS

3-1. GENERAL PROGRAM

3-2. Upon solution, a General Program problem is placed on a program sheet (Coding and Constant Sheet), punched on paper tape, and written onto the General Program section of the magnetic drum.

3-3. Coding and Constant Sheets. The coding and constant sheets are used by programmers for general purpose programs. These program sheets are broken up into 13 individual columns as follows:

Column 1 - INSTRUCTION NUMBER - Used to count the number of instructions in a particular program. Decimal numbers are used in this column.

Column 2 - MNEMONIC CODE - Abbreviation for the instruction being used (i.e., Multiply instruction would be listed in the Mnemonic Code Column as MLT).

Column 3 - O.P. CODE - Instruction number (MLT Instruction is 03). Octal numbers only are used in this column.

Column 4 - MEM. ADD. - Memory address of the instruction to be used. Octal numbers only are used in this column.

Column 5 - SCALE - Scaling of the number in the Arithmetic accumulators. Scaling is to powers of two only.

Column 6 - ARITHMETIC ACCUMULATOR - The number actually being operated on. This number is in binary format.

Column 7 - SALV. - Salvage register column. Stores the contents of the accumulator after a load instruction.

Column 8 - BOOLEAN ACCUMULATOR - Serves the same purpose as the arithmetic accumulator. Used for Boolean instructions only.

Columns 9-12 - SALV. 1 - SALV. 4 - Boolean salvage registers. Serves the same purpose as column 7. Bits may shift from one register to another.

Column 13 - REMARKS - Used for brief explanation of instruction where necessary.

3-4. The completed program on the Coding and Constants Sheets is punched on paper tape by an electrical keyboard. The keyboard, which resembles a conventional desk calculator, eliminates the need for punched card inputs to the tape preparation unit.

3-5. Data Format. All instructions will be written into the keyboard using six octal digits: Most Significant Code digit first, second code digit, then address, Most Significant Digit first.

3-6. No-Address Load Instructions (LDK), will require a keyboard entry of Octal "22" to activate a fractional decimal-to-octal translator. The four digit fractional decimal number will then follow, being entered through the keyboard.

3-7. Boolean Instructions will have the address portion tabulated 0000 to 3777 (Octal), which represents 128 16-bit words. All remaining instructions will be written as two-digit code and four-digit address or control word.

3-8. An example of a General Program Computer Word and binary equivalent is shown in figure 3-1 where dn represents each octal digit, "0" stands for binary bits not recognized, and "X" represents valid binary bits.

d_1	d_2	d_3	d_4	d_5	d_6
OXX	XXX	OXX	XXX	XXX	XXX

Figure 3-1. General Program Computer Word

3-9. Each General Program Computer Word occupies two lines of punched paper tape, each line capable of holding eight binary digits. A punched holes in the tape signifies a one, and a blank, zero. Tape format for a General Program Computer Word is illustrated in figure 3-2, where "C" is the O.P. Code and "A" is the address.

C C C C C • A A A

A A A A A • A A A

Figure 3-2. Tape Format, General Program Computer Word

3-10. A sample General Program forming the equation I = E/R is illustrated in figure 3 3. Part (A) is the formation of the equation on a "Coding and Constant" sheet, and Part (B) is the same program on paper tape.

DATE APRIL 23, 1963

PROBLEM CURRENT (I) COMPUTATION

PROGRAMMER JOHN DOE

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-2214-A

PAGE 1 OF 1

INSTRUCTION NUMBER (1 2 3 4 5 6 7)	MNEMONIC CODE	O.P. CODE (8 9 10)	MEM. ADD. (11 12 13 14 15)	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
0 1	LD	2 0	1 2 0 6		E							Load E Into the Arithmetic Accumulator
0 2	DIV	0 6	2 4 1 0		E							Divide the Contents of the Accumulator By R
0 3	NPA	1 2	0 0 0 0		E							Instructions 03-06 Used to Give Computer Time To
0 4	NPB	2 5	3 7 7 7		E							Complete Divide Instruction
0 5	NPA	1 2	0 0 0 0		E							
0 6	NPB	2 5	3 7 7 7		E/R							
0 7	ST	2 3	0 7 7 7		I = E/R							Store the Contents of the Accumulator in the Memory Location of I.

Figure 3-3. General Program Part A - (Coding and Constant Sheet)

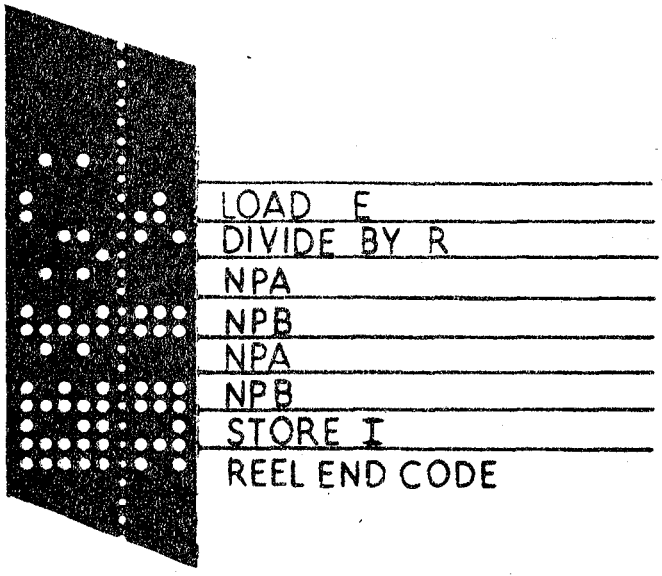


Figure 3-3. General Program Part B - (Punched Paper Tape)

3-11. INTERPOLATOR PROGRAM.

3-12. Solutions of flight equations are placed on "Linear Function Interpolator Data Input Sheets", punched sequentially on paper tape, and put into the interpolator bands of the magnetic drum.

3-13. Linear Function Interpolator Data Sheet. The Linear Function Interpolator Data Sheet is divided into two sections. The first section contains information concerning the number of variables, whether the variables are indexed scaling of variables, etc. The second section contains the location of the breakpoints. The first section is further broken-down into cards 1, 2, and 3. This is done for convenience where punched cards and a card reader is used in conjunction with the tape Preparation Unit.

3-14. A standardized procedure is required to fill out the LFI input sheets to maintain full usage and uniformity in all interpolator programs. The standard form of each item on the LFI input sheet is presented as follows:

CARD 1

Function Numbers - the function number space has twelve blocks assigned. The first three blocks will be used to show the numerical function number. The second three blocks are used for the first variable input number, the third set of three blocks for the second variable

input number, and the last three blocks for the third variable input number. Function numbers are arbitrarily assigned by the programmer. These numbers may be written either decimally or octally on the input sheet. An example of a three variable function number follows:

f40 $M^{(12)}$, $\alpha e^{(11)}$, $\alpha se^{(11)}$ would be written as
040002 009022 decimally or
050002 011026 octally where the input number for $M^{(12)}$ is 2, $\alpha e^{(11)}$ is 9, and $\alpha se^{(11)}$ is 22. The function number used is 40.

The programmer should be consistent when writing the function number. If the decimal numbering system is used for the function number in the rest of the interpolation program, it should be used in any new programs. The same holds true for octal numbers.

Sort Number - This number is any convenient combination of alphabetics and numerics which describes the order of this card. Two alphabetics and three numerics will be used. (This is only used with punched cards and card reader.)

CARD 2

X Address - (X ADD) - The address will contain the core address (in octal) of the independent variable X. If the term is to be indexed (the same set of curves and with different inputs), the X ADD will be the core address (in octal) of the first item to be indexed and must have a zero or a four as the last numeric in the octal number. The next three serial octal core locations must be the remaining indexed inputs. Core location 145 will appear as 0145.

Repeat X (RPX) - The number of times the X address will appear in the output data. If the X address is to appear five times, this will be written on the sheet as 05. As the present, five will always be used.

Index X (IXX) - If X is to be indexed, X will appear. If X is not to be indexed, the space will be left blank.

Y Address (Y ADD) - Similar to X address except this is for the second variable in a two or three variable function. For a single variable this will be left blank.

Repeat Y (RPY) - The number of times the Y address is to appear in the output data. Five is the only number used at this time, and it would appear on the input sheet as 05. These blocks are only used for a function of two or three variables.

Index Y (IXY) - If Y is to be indexed, Y will appear. This will be filled out only if this is a function of two or three variables.

Z Address (Z ADD) - Similar to X ADD except for the third variable of a three variable function. For single or double variable functions this will be left blank.

Repeat Z (RPZ) - The number of times the Z address is to appear in the output data. Five is the only number used at this time and would appear on the input sheet as 05. This will be filled out only for a three variable function.

Index Z (IXZ) - If Z is to be indexed, Z will appear to be filled out only for a function of three variables.

Sort Number - Number used to signify the order of this card in the input data.

CARD 3

Answer Address (A ADD) - The answer address will contain the core address (in octal) where the answer (dependent variable) will be located. For functions which are being indexed, the answer address will be that of the first answer and the core location in octal must have a zero or a four in its last numeric. The next three core locations will be the indexed answers.

Repeat answer (RPA) - The number of times the answer is to appear in the output data. Five is the value of the number to be used and will appear on the input sheet as 05.

Computer time (CT) - The number of words consisting of all zeros that is to appear in the output data. 02, 04, or 06 will be used for a function of one, two, or three variables, respectively.

Decimal Point (DP) - The number of places the decimal point in the input data is to be shifted. The decimal point in the input data is always at the extreme left. 50 written in these blocks implies no shift, 51 implies a decimal point shift to the right of one place, 52 means a shift to the right of two places, 49 means a shift to the left of one place, 48 shift left two places, etc.

Scale (SC) - The proper power of two. -05 implies that the adjusted decimal number is to be divided by 32, +07 means the adjusted decimal number is to be divided by 0.0078125. As an example, if the largest decimal number for this function is 15.87, the scale factor is -04; or if the largest decimal number was 0.007753, the scale factor would be +07.

Sort Number - Signifies the order of this card in the resultant card deck.

The lower half of the input sheet contains 81 blocks for the independent variables. Since a single variable has nine data words (f0, f1/8, f1/4, f3/8....f1.0), a single page may contain all the data words for a two variable function (9 x 9 = 81). A three variable function would have 729 data words (9 x 9 x 9) and require nine input sheets to show the whole program.

3-15. Information contained in this section is in decimal form. Prior to being punched on paper tape, it must be divided by the appropriate scale number and converted into octal numbers.

3-16. Data on the Linear Function Interpolator Data Input Sheets is punched sequentially on paper tape using an electric keyboard which is part of the Tape Preparation Unit.

3-17. Interpolator Words. Interpolator words written into the keyboard are divided up into instruction words and data words.

a. Interpolator instructions are written into the keyboard as follows:

(1) When a flag bit is written in the most significant bit position (bit 11), a separate manual key (F) shall be operated prior to writing the instuction.

(2) For a control instruction word, four octal digits shall be written: Zero; variable value 1-3; zero; variable value 0-7. Largest number possible would be 0307.

(3) For an address instruction word, four octal digits shall be written: most significant digit first, with values: 0001-1777. The binary equivalent of interpolator instruction words is shown in figure 3-4, where F is the flag bit, dn are octal digits, 0 is a bit not recognized and X is a valid binary bit.

	d_1	d_2	d_3	d_4
F	00X	XXX	XXX	XXX
	1	7	7	7

Figure 3-4. Interpolator Instruction Word

b. Interpolator data words shall be written as follows:

(1) The flag bit shall be written in the identical manner as the interpolator instruction flag bit and shall appear with the first data word only.

(2) The data words shall be written with four decimal digits, in fractional decimal form, the most significant digit first, with values: .0000 - .9999. These data words are internally translated to a ten-bit fractional binary equivalent, without roundoff. The binary/octal equivalent is illustrated in figure 3-5, where F is the flag bit, dn are octal digits, 0 is a bit not recognized and X is a valid bit.

	d_1	d_2	d_3	d_4
F	XXX	XXX	XXX	X00
	7	7	7	4

Figure 3-5. Interpolator Data Words

3-18. Interpolator Tape. An interpolator program is divided into several zones on paper tape. The least number of zones allowed is six for a single variable function, seven for two variable functions, and eight for three variables. These zones are as follows:

Zone 1 - Control Word - One word, the first bit of which is always one (flag bit) and the second bit always zero. The tape format for a control word is shown in figure 3-6.

D D D D F . 0 0 X

X 0 0 0 X . X X 0

Figure 3-6. Control Word Tape Format

D represents the dummy code which is always 0111 and is located at the beginning of every interpolator word. F is the flag bit which is always 1 in the control word, 0 is bits not recognized and X is valid bits. The binary/octal format of the control word is shown in figure 3-7, where octal word A is the flag bit, octal digit C is not used, and octal digits B and D are coded words. Octal digit B tells whether the function is of one, two or three variables, D tells which of the variables are indexed.

A	B	C	D
F0	XXX	000	XXX

Figure 3-7. Control Word Binary/Octal Format

Octal digit B is coded as shown in table 3-1.

Table 3-1. Control Word, Octal Digit B Code

Number of Variables	Binary Equivalent
1	001
2	010
3	011

Octal digit D is coded as shown in table 3-2.

Table 3-2. Control Word, Octal Digit D Code

Indexed Variables	Binary Equivalent
None	000
X	001
Y	010
X, Y	011
Z	100
X, Z	101
Y, Z	110
X, Y, Z	111

Zone 2 - Consists of five identical words. It is the core memory location of the independent variable X. The first word in zone 2 always has a flag bit and all five words have the dummy code. The binary equivalent of zone 2 words is shown in figure 3-4.

Zone 3 - Identical to zone 2 except that it contains the address of the independent variable Y.

Zone 4 - Identical to zone 2 or zone 3 except that it contains the address of the independent variable Z.

Zone 5 - The data field consisting of the sequentially stored data words which describe the function at fixed breakpoints. The first word in this zone always contains a flag bit, and all words have a dummy code. The binary equivalent for the words contained in this zone is shown in figure 3-5.

Zone 6 - Blank words used to allow the computer time to finish the interpolation. The first word in the zone always contains a flag bit (F) and all words contain a dummy code (D). Tape format for a computer time word is illustrated in figure 3-8.

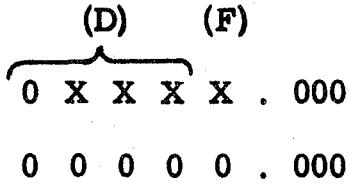


Figure 3-8. Computer Time Word, Tape Format

Zone 7 - Contains the core memory address where the result of the interpolation f(X, Y, Z) is to be stored. The address is repeated four times. The first word always contains a flag bit and all four words have a dummy code. The binary equivalent of the answer word is illustrated in figure 3-4.

Zone 0 - The same as zone 7 except it is a single word always containing a flag bit and dummy code. Zone 0 is used to reset the counter, which tells the computer that the interpolator program has been completed.

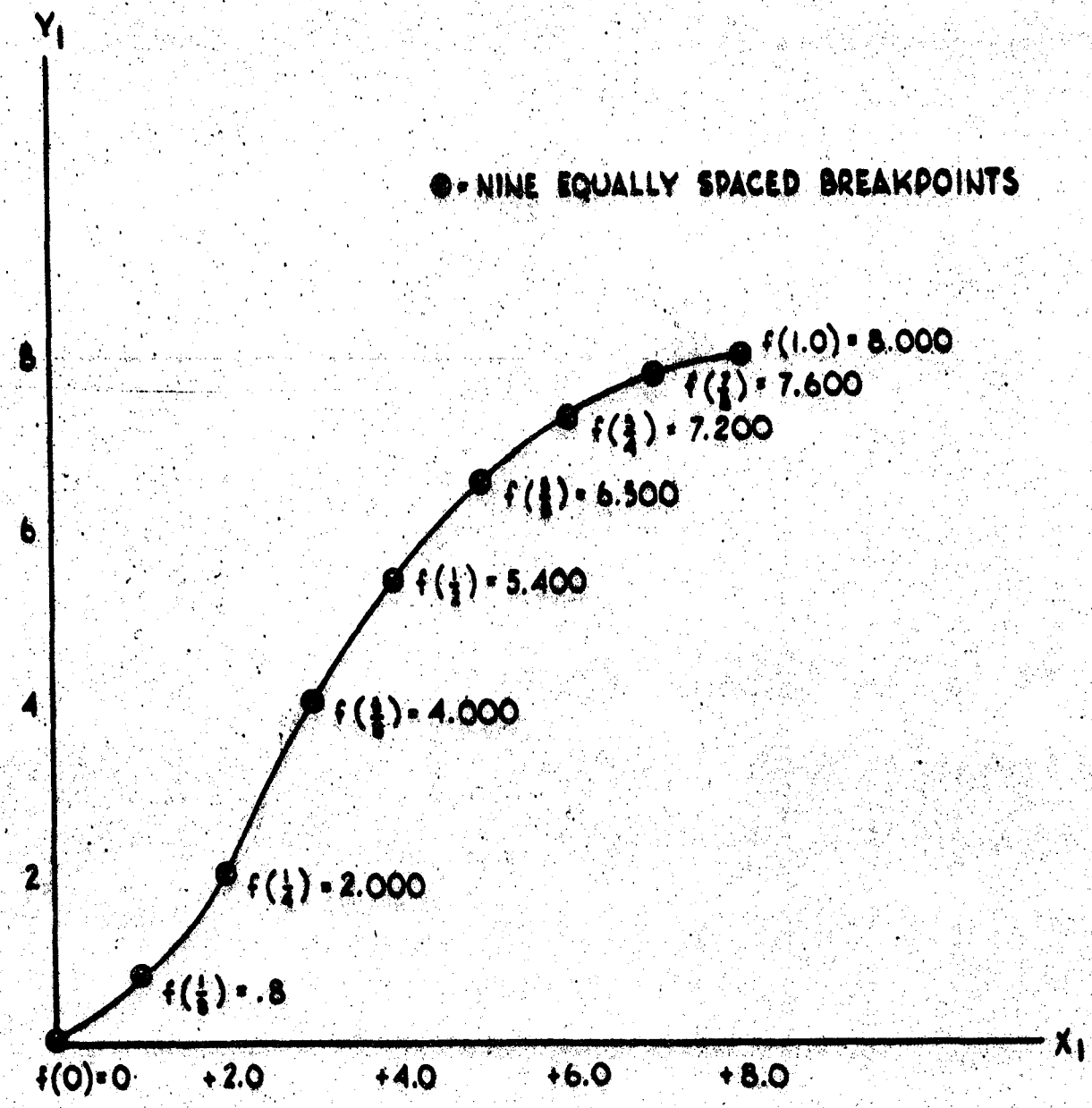


Figure 3-9. Interpolator Program Part A - Function Graph

DATE 6-26-63 | LINEAR FUNCTION INTERPOLATOR DATA INPUT SHEET | PAGE NO. 1

REV.-- | REP.NO.

CARD (1) FUNCTION NUMBER 009002 AA001 SORT NO.

CARD (2) X ADD 1123 RPX 05 IXX Y ADD RPY

IXY Z ADD RPZ IXZ SORT NO.

CARD (3) A ADD 1576 RPA 05 CT 02 DP 51 SC -03

SORT NO.

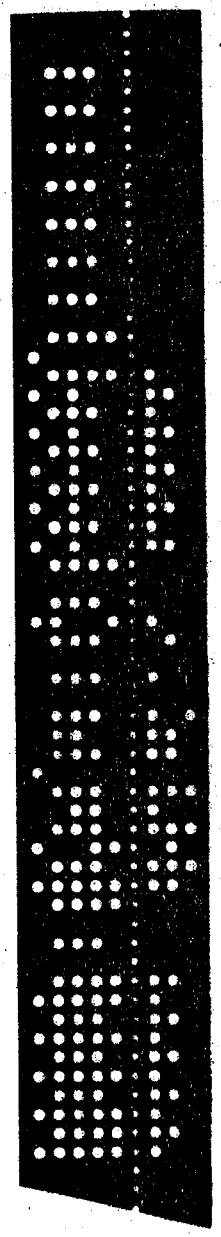
CODE	X	
1	0	0000
2	1.0	0800
3	2.0	2000
4	3.0	4000
5	4.0	5400
6	5.0	6500
7	6.0	7200
8	7.0	7600
9	8.0	8000

CARD NO.

REFERENCE:

NO.	CHANGE	NAME	DATE

Figure 3-9. Interpolator (Part B - Data Input Sheet)



TAPE LEADER

7 DUMMY CODES

CONTROL WORD - ZONE 1

X ADDRESS - .1123
ZONE 2

DATA WORDS
ZONE 5

- $f(0) = .0000$
- $f(\frac{1}{8}) = .0630$
- $f(\frac{1}{4}) = .2000$
- $f(\frac{3}{8}) = .4000$
- $f(\frac{1}{2}) = .5314$
- $f(\frac{5}{8}) = .6400$
- $f(\frac{3}{4}) = .7150$
- $f(\frac{7}{8}) = .7164$
- $f(1.0) = .7774$

COMPUTER TIME - ZONE 6

ANSWER ADDRESS - 1576
ZONE 7

ANSWER ADDRESS - ZONE 0

Figure 3-9. Interpolator Program Part C - Punched Paper Tape

3-19. A sample interpolator program involving a one variable functions is as follows: Figure 3-9 Part (A) illustrates a graph of a certain function; Part (B), the data which would be contained on the Linear Function Interpolator Data Input Sheet; Part (C) the corresponding program punched on paper tape.

3-20. RADIO AIDS PROGRAM.

3-21. Information pertaining to Radio Aids preselection is placed on a Radio Facility Data Sheet punched on paper tape and loaded onto the Radio Aids band of magnetic drum.

3-22. Radio Facility Data Sheet. The Radio Facility Data Sheet is divided into three sections, Card #1, Card #2 and Card #3. The breakdown of the sheet into card numbers is for convenience where punched cards and a card reader are used in conjunction with the tape preparation unit.

3-23. A standard procedure is required to fill out the Radio Facility Data Sheets to maintain full usage and uniformity in all Radio Aids programs. The procedure for completing the Radio Facility Data Sheet follows:

Card #1

ALL-TYPE-Six characters - fill in starting at extreme left. Any unused spaces will appear on the right. This will be filled in for all types of Radio Aids.

Example: ILS would appear as ILS

ALL-CALL LETTERS - Seven characters - fill in starting at extreme left. Any unused spaces will appear on the right. This will be filled in for all type Radio Aids. Two or three alphabetics will usually be written, but for FM, BONE, Z, and LFM, numerics will be written. A three will represent a dash and a one will represent a dot.

Example: a. ACY would appear as A C Y.
b. .-.- would appear as 13.13.

ALL-LATITUDE (Point of Touchdown) - Seven digits - one digit for sign, two digits for degrees, and four digits for fractional parts of a degree. N4.2256 degrees would appear as +04 2256. This will be filled in for all types of Radio Aids.

ALL-LONGITUDE (Point of Touchdown) - Eight digits - one digit for sign, three digits for degrees, and four digits for fractional parts of a degree. E74.8279 degrees would appear as +074 8279. This will be filled in for all types of Radio Aids.

ALL-ELEVATION (Feet) - five digits, 518 feet would appear as 00518. This will be filled in for all types of Radio Aids.

LF, LFRR-POWER - four digits. 512/2048 would appear as 0512. 2048's considered max power equivalent to 480 NM range. Thus 512 = 120 NM or 1/4 MAX RGE. This is filled in for types 6 and 7. (See Appendix A.) The type is the most significant digit of the sort number.

ALL-FREQUENCY - (MC or KC) - four digits filled in for all types.

a. 400 KC would appear as 0400, for types 6 and 7, but for types 4 and 5 the decimal point will be after the third significant digit. 117.5 would appear as 1175.

b. A special code will be used for types 1, 2, and 3.

   (1) Zero to full scale is 0700.

   (2) One-half to full scale is 2700.

   (3) Zero to one-half scale is 0300.

VHF, LFRR-COSINE MAG. VAR. (Cosine magnetic variation) - four decimal digits. Filled in for types 5 and 7.

ILS-RL-RTG +1000 (runway length minus distance from threshold to touchdown plus 1000 feet). This represents the distance between TD and Localizer in feet - 5 digits, 9971 would appear as 09971. Filled in for type 4 only.

ALL-COSINE LATITUDE - Cosine of the latitude in the input data; four digits with decimal point at extreme left. Filled in for all types.

ALL-SORT NUMBER - any convenient combination of alphabetics or numerics that describe the order of this card (where used). This number does not enter in the computation. Filled in for all types.

Example: 6 NY BGM 1. In this sample 6 represents the type facility, NY the state, BGM the call letters, and 1 the card number.

Card #2

MARKERS - SINE AXIS - sign and four decimal digits. Axis of facility is measured positive clockwise from north (negative measured counterclockwise from north). If the angle is between 0 and $+90^{o}$, sign of the sine is positive. If the angle is between 0 and -90^{o}, the sign of the sine is

negative. Decimal point of the sine is always assumed to be at the extreme left, +.0156 would appear on the form as +0156. Filled in for types 1, 2 and 3.

b. COSINE AXIS - Sign and four decimal digits. Axis is measured positive clockwise from north. Sign of the cosine is always positive. The decimal point of the cosine is always assumed at the extreme left. +.0156 would appear as +0156. Filled in for types 1, 2 and 3.

VHF - a. MAG. VAR. (Magnetic Variation) - write in directly.

b. SINE MAG. VAR. - five digits, one digit for sign, four digits for the decimal value of the sine of the magnetic variation. An east magnetic variation would have a plus sign, and a west magnetic variation would have a minus sign.

ILS - Cross out facilities not associated with the ILS installation.

a. RNWY BRG (MAG) - Magnetic runway bearing in degrees with 360° magnetic north. Write in directly.

b. MAG VAR - Magnetic variation. Write in directly.

c. RNWY BRG (TRUE) - True runway bearing. Write in directly. (TRUE = MAG + M.V.) + for East, - for West.

d. SINE BRG - Sine of the true runway bearing. One digit for the sign of the sine and four digits for the numerical value of the sine.

e. COSINE BRG - Same as d. except cosine.

f. G/S - Glide Slope Angle (degrees) - three digits. The decimal point is assumed to be after the first digit. 4.11 degrees would appear on the input sheet as 411. Filled in for type 4. Maximum angle 4.99 degrees, minimum 2.00 degrees.

LFRR - a. MAG. VAR. - Magnetic variation - same as previously explained.

b. SINE AXIS - Five bits, one bit for sign and four bits for magnitude.

c. COSINE AXIS - Five bits, one bit for sine and four bits for magnitude.

d. LEG NO. - MAG. BRG and DIA

(1) LEG NO. - The four A-N course legs. The leg numbers are counted clockwise starting in the northerly quadrant.

(2) MAG. BRG - Magnetic Bearing, write in directly.

(3) DIA - Diameter of the circular radiation patterns. Consist of four decimal digits with the decimal point at the extreme left.

ALL - a. MISC - Miscellaneous - six bits to be used whenever necessary (i.e., simulating certain foreign facilities). If none used, leave blank.

b. SORT NO. - Same as Card #1.

Card #3

SIMULATOR NO. - any convenient seven character array that describes the simulator being worked on. This number does not enter into the computation.

BASE ADDRESS, J NUMBER - base address will be a four digit number and J will be a two digit number. The base address is the octal drum address of the start of the preselect data for this facility, and the J number is an octal number ranging from 00 to 37. A base address of 540 and a J number of 7 will appear on the sheet as 0540 07. Filled in for all types.

SORT NO. - Same as Card #1.

3-24. Processing Type Code 1 and 2 Inputs (1 = MM, 2 = OM). Each card of output for the radio aids will contain up to a maximum of seven character words. Each output card, in addition to the required numerical octal information, will contain the following:

a. Call letters - max. seven alphabetics or numerics.

b. Type - max. six alphabetics.

c. Card number - two numerics.

d. Octal address - four numerics.

e. Simulator - max seven alphabetics or numerics.

This output data will be punched out from what appears in the input data. The remaining two words on the card will be occupied by octal information.

3-25. Preselect Data - The preselect data will be placed on punched cards as follows. There is one word punched on each card, therefore Card 1 = Word 1, etc.

Card 1. The first two octal words of the preselect data for outer marker or middle marker inputs will consist of the data listed below. Items a, b, c, d and e will be punched on the card. The remaining two words will be all zeros.

At least one space will separate the two octal words. Call letters, type, and simulator number will not vary from output card to output card for each station, but card number and addresses will. For this card, the card number will be 1 and the address (ADDRESS) will equal the base address in the input data.

Card 2. The second card to be punched will consist of the same data that was punched on the first card, except the address will be incremented by one (octal), and the card number will be 2. The two octal words will be 000077 330000 and 200077 330000 or 000037 330000, depending upon the values in the frequency input field. (See table 3-3.)

Table 3-3. Input Frequency versus Octal Output

Input Card Frequency	Octal Output
0700	000077 330000
2700	200077 330000
0300	000037 330000

Card 3. The third card to be punched will be similar to Card 2 except the address will be incremented by one and the card number will be 3. The two octal data words will consist of latitude lower and latitude upper, which will be formed as follows:

a. The latitude lower and upper will be formed respectively by subtracting the size to lower latitude from the latitude, and by adding the size to upper latitude to the latitude. Table 3-4 gives the size to upper and lower latitude, longitude, and frequency.

Table 3-4. Size to Upper and Lower Limits

Type	Latitude (Degrees)	Longitude (Degrees)	Frequency (KC)
OM, MM, Z, LFM	±0.1000	±0.1000	Not applicable
BONE, FM	±0.6667	±0.6667	Not applicable
ILS, ILSDME	±1.0	±1.0	Not applicable
MVRTAC, TACAN, UHF-DF, LVR HVR, MVR, HVRTAC	$\frac{\text{Power}}{258}$	$\frac{\text{Power}}{256}$	Not applicable
LOM, LMM, RBN, BC, LFRR	$\frac{\text{Power}}{256}$	$\frac{\text{Power}}{256}$	06

b. The two octal words will be formed in the following manner. The lower latitude will be divided by 256 to produce a seven digit decimal number from 0000 to .9999999. This number will be multiplied successively by two, to produce a ten bit binary number which are the ten most significant bits of the resultant 20 bit number. After division by 256, the upper latitude will form the ten least significant bits. The resultant 20 bit binary word will be converted to two octal words as follows:

Octal Word 1

a. 1st character - beginning of word definer

b. 1st two significant bits - form octal digit 0 to 3

c. next three bits - form octal digit 0 to 7

d. next two bits - form octal digit 0 to 3

e. next three bits - form octal digit 0 to 7

f. next three bits - form octal digit 0 to 7

g. next three bits - form octal digit 0 to 7

Octal Word 2

h. next two bits - form octal digit 0 to 3

i. next two bits - form octal digit 0 to 3

Four zeros will make up the last four digits of the second octal word. This condition is true for the second octal word of all data.

Card 4. The same process that was used on Card 3 can be used to form Card 4 except the sizes will be divided by cos latitude before they are subtracted or added to longitude. The lower and upper longitude will be formed with a 20 bit number and converted to octal in the same manner employed with Card 3. Again, the base address will be incremented by one and the card number will be 4.

3-26. Data Words. There are four data words for the outer or middle markers.

a. Data Word A - latitude - 19 bits, with a leading sign bit (0 = plus, 1 = minus)

b. Data Word B - longitude - 19 bits, with a leading sign bit (0 = plus, 1 = minus)

c. Data Word C - sine (10 bits) cosine (10 bits)

d. Data Word D - ten leading zeros and elevation +1000 feet (10 bits)

3-27. Each word will be defined by two octal words, as was the preselect data. The call letters, type, card number, base address and simulator will also be punched on the card.

Card 5. The call letters, type and simulator will be punched the same way as in Cards 1, 2, 3, and 4. Two hundred will be added octally to the base address +J to form the octal address for this card, and the card number will be five. The two octal words representing latitude will be formed in the following manner: The latitude, which was stored when the input data was read in, will be divided by 256 and converted to a 21 bit pure binary number by successive multiplications by two. The first two most significant bits will be ignored, the most significant bit will be made a zero, and the 20 resultant bits will be converted to octal in the same manner as Card 3.

Card 6. The same as Card 5 except that longitude will be converted to two octal words after division by 256. The most significant bit will be made a

one instead of a zero as in Card 5. The octal address will be formed by octally adding 40 to the octal address of Card 5, and the card number will be six.

Card 7. The card number is seven. The address is octal 40 plus the octal address of Card 6. The sine and cosine of the axis which is contained in the input data will be converted to binary and then to octal in the following manner: the sign of the sine will form the most significant bit. The remaining nine bits of the sine will be formed by multiplying by two, nine times. The cosine will be treated in the same manner to form the ten least significant bits of the 20 bit word. The 20 bits will be converted to two octal words forming data word C.

Card 8. The card number is eight, and the octal address will be octal 40 plus the octal address of Card 7. The 20 bit elevation word will be formed of the six miscellaneous bits and four zeros, dividing the elevation +1000 feet by 16,384, and multiplying by two ten times to produce the remaining ten bits. The 20 bits will be converted to two octal words forming data word D.

3-28. Processing Type Code 3 Inputs (Fan Z Marker). This program will be used to process FM, Z, BONE or LFM. The input data is the same as that used with type Codes 1 and 2.

3-29. Preselect Data. Formation of the preselect data will be exactly the same technique used for type Codes 1 and 2.

3-30. Data Words. Data words A, B, and C will be formed in the same manner as type Codes 1 and 2. In the case of data word D, instead of the first ten bits of the 20 bit word containing six miscellaneous bits and four zeros, the first four bits will be formed according to the type of facility as shown in table 3-5.

Table 3-5. First Four Bits of Data Word D (Type Code 3)

Type	Binary Bits
LFM	1000
BONE	0100
FM	0010
Z	0001

3-31. The remaining six bits of the ten most significant bits of data word D will be "padded" with zeros. The ten least significant bits will contain elevation and will be formed in exactly the same manner as the ten bits representing elevation in Type Codes 1 and 2.

3-32. Call Letter Generation. Call letters must be generated in this program in addition to the preselect and data words. There are four 20 bit words of call letters, although only the 16 most significant bits of each word will be used. The remaining four bits will be miscellaneous bits and/or zeros.

3-33. When a Z marker facility is to be simulated, 16 ones would be used, making the first octal word for Cards, 9, 10, 11 and 12 equal to 373777.

Card 9. This card will contain the four most significant MISC bits. They will form binary bits 17, 18, 19 and 20. These are converted to be the first two octal bits of Card 9's second octal word. The remaining four octal bits will be zeros.

Card 10. This card will contain the two least significant MISC bits which will form binary bits 17 and 18. These bits are converted to the first octal bits of Card 10's second octal word. The remaining five octal bits are zeros.

Card 11 and Card 12 will have their second octal word equal to 000000.

3-34. If any of the other types of fan markers (FM, BONE or LFM), are to be simulated, the call letter words must be generated from the numerical information contained in the input data word representing the call letters. (A three will produce three ones followed by a zero. A one will produce one 1 followed by a zero.) The program will continue to follow this process and count the number of bits it has formed starting with the most significant digit. A one will be subtracted from the counter after the last numeric is processed.

3-35. When all the bits are formed, the computer will divide the count into 64 to determine the quotient and the remainder.

a. If the quotient is less than or equal to the remainder, the remainder will be divided by the quotient, and the integer obtained will be the number of zeros placed between the call letters and after the last call letter. The call letters will be placed in the 64 bits the number of times specified by the quotient.

b. If the quotient is greater than the remainder, the quotient will be reduced by one, and the count will be added to the remainder.

c. If the reduced quotient multiplied by five is less than or equal to the accumulated remainder, the remainder will be divided by the quotient. The integral result will be the number of zeros inserted between the call letters in the 16 bit words. Any uneven number of zeros left over will appear at the end of the last call letter.

d. If the reduced quotient multiplied by five is greater than the remainder, repeat steps b and c.

3-36. The 64 bits will form 16 bits of four binary words. The four least significant bits (17, 18, 19 and 20) of the first word will be the four most significant MISC bits. The 17th and 18th bits of the second word will be the two least significant MISC bits. All remaining bits needed to form four 20 bit words will be zeros.

3-37. Processing Type Code 4 Inputs (ILS).

3-38. Preselect Data. The preselect data will consist of two octal words consisting of zeros on the first output card. The second output card will consist of frequency in the first ten bits. The first nine will be as shown in table 3-6. These same nine bits will be the 11 through 19 bits inclusive of the frequency word. Bits 10 and 20 are illustrated in table 3-7.

Table 3-6. Type Code 4 Frequency Bits 1 thru 9

Frequency	Bits 1 thru 5	Frequency	Bits 6 thru 9
108	10100	.0	1000
109	10101	.1	0100
110	11010	.2	1010
111	11011	.3	0101
112	11100	.4	0010
113	11101	.5	1001
114	01110	.6	1100
115	01111	.7	0110
116	00110	.8	0011
117	00111	.9	0001

Table 3-6. Type Code 4 Frequency Bits 1 thru 9 (Cont)

Frequency	Bits 1 thru 5	Frequency	Bits 6 thru 9
133	10111		
134	01010		
135	01011		

Table 3-7. Type Code 4 Frequency Bits 10 and 20

Type	Bit 10	Bit 20
ILS	0	0
ILS DME	0	1

Cards 3 and 4 will consist of latitude and longitude lower and upper.

3-39. Data Words. Cards 5 and 6 will consist of latitude and longitude (20 bits each), formed in the same manner as previously described.

Card 7 will consist of true runway bearing sin and cos; one bit for sign and nine bits for the magnitude of both sine and cosine for a total of 20 bits. A zero in the sign position denotes positive sign and a one denotes negative sign.

Card 8 will consist of one and zero in the two most significant bits for ILS and two ones for ILS DME. The next six significant bits will be the MISC. bits from the input data. The 12 least significant bits (for a total of 20 bits) will be elevation. The elevation +1000 in the input data will be divided by 13,384 and converted to 12 bit binary.

3-40. Call Letters. Cards 9, 10, 11 and 12 will consist of the call letters. The call letters will be formed once from the alphabetic call letters in the input data and not repeated, as in the case with markers. The call letter bits will be formed starting with the first word and any unused bits will be "padded" with zeros. Spaces between letters will be represented by three zeros as shown in table 3-8. There will be a maximum of three call letters in the input data.

Table 3-8. Call Letter Generation

Call Letter	Binary Equivalent	Number of Bits
A	10111000	8
B	111010101000	12
C	11101011101000	14
D	1110101000	10
E	1000	4
F	101011101000	12
G	111011101000	12
H	1010101000	10
I	101000	6
J	1011101110111000	16
K	111010111000	12
L	101110101000	12
M	1110111000	10
N	11101000	8
O	11101110111000	14
P	10111011101000	14
Q	1110111010111000	16
R	1011101000	10
S	10101000	8
T	111000	6
U	1010111000	10
V	101010111000	12

Table 3-8. Call Letter Generation (Cont)

Call Letter	Binary Equivalent	Number of Bits
W	101110111000	12
X	11101010111000	14
Y	1110101110111000	16
Z	11101110101000	14

3-41. The four least significant bits of Cards 9, 10, 11 and 12 will consist of the binary representation of glide path angle, and runway length minus distance from threshold to touchdown. Two will be subtracted from the glide path angle in the input data and the result will be divided by three and converted to binary. The four most significant bits of glide path angle will be placed in the four least significant bits (17, 18, 19 and 20) of Card 9. The four least significant bits of glide path angle will be placed in the four least significant bits of Card 10.

3-42. The eight binary bits of runway length minus threshold to touchdown +1000 feet after division by 16,384 will be placed in the four least significant bits of the data word on Cards 11 and 12. The four most significant bits will be described by the second octal word of Card 11 and the four least significant bits will be described by the second octal word of Card 12.

3-43. Processing Type Code 6 Inputs (LF).

3-44. Preselect Data. The preselect data will consist of two octal words consisting of zeros on the first output card. The second card will consist of the lower frequency in the first ten most significant bits and upper frequency in the ten least significant bits. Frequency will be formed by subtracting the size (table 3-4) to lower frequency from the frequency in the input data, and adding the size to upper frequency, to the frequency in the input data. The two formed numbers will be divided by 2048 and converted to ten bit binary. Cards 3 and 4 latitude and longitude will be formed as previously described.

3-45. Data Words. Cards 5 and 6 will be 20 bit latitude and longitude respectively. Card 7 will be the frequency converted to binary after division by 2048. The 13 most significant bits will contain frequency (with a leading bit of zero), and the seven least significant bits will be padded with zeros. Card 8 will contain a leading bit of zero. The next nine significant bits will contain power (divided by 2048) and converted to nine bit binary. The last ten bits (Card 8) will contain elevation +1000, formed as previously described.

3-46. Call Letters. The alphabetic call letters contained in the input data will be used to generate the 64 bits of call letters. The call letters will be generated according to table 3-8. Call letters will be generated as many times as possible for one letter calls, twice for two letter calls, with at least five zeros spacing between each word, and after the last call letter. Where three call letters are used, generate the call only once.

3-47. The process to be used is explained in paragraph 3-34 to 3-39. The four least significant bits in each of the third and fourth call letter words will be padded with zeros. The first and second call letter words will have the MISC. bits.

3-48. Processing Type Code 7 Inputs (LFRR).

3-49. Preselect Data. The preselect data words are formed in exactly the same manner as used in Type Code 6 (LF).

3-50. Data Words A, B, C, and D.

Data Word A - Card 5 - 20 bit latitude

Data Word B - Card 6 - 20 bit longitude

Data Word C - Card 7 - The most significant 13 bits (with a leading bit of zero), contain the frequency. The seven least significant bits are "padded" with zeros.

Data Word D - Card 8 - Most significant bit = 1, next nine bits = power divided by 2048, ten least significant bits = elevation +1000 divided by 16,384.

3-51. Call Letters. Same as Type Code 4 process, paragraph 3-42 except the first 16 bits of the first call letter word will be zero. Remainder of bits will contain the call letters formulated once, with zero "padding" for the remaining bits.

3-52. Data Words E, F, G, and H. These data words are used only in the simulation of LFRR facilities. The storage locations of these words can be found by utilizing table 3-11.

3-53. Data Word E punched on card 13 will consist of the diameters of the first two circles (LEG NO. 1 and 2), in the input data (ten bits each). Data word F punched on card 14 consists of the diameters of the last two circles (LEG NO. 3 and 4), in the input data (ten bits each).

3-54. Data words G and H, cards 15 and 16, will consist respectively of the sine and cosine of the axis, and the sine and cosine of the magnetic variation. The sine will be made up of one bit for sign and nine bits for magnitude. The cosine will always be positive and consist of a zero and nine magnitude bits. These conditions hold true for both axis and magnetic variation.

3-55. Processing Type Code 5 Inputs (UHF/VHF).

3-56. Preselect Data. The first preselect word will contain 20 bits of zeros. The second preselect word is frequency which is formed from table 3-6 except that the 10th and 20th bits are as shown in table 3-9.

Table 3-9. Type Code 5 Frequency Bits 10 and 20

Type Facility	Tenth Bit	Twentieth Bit
-VR	0	0
-VR - TC	0	1
-VRDME	0	1
-TC	1	1

3-57. The third and fourth preselect words will be two 20 bit words consisting of latitude and longitude lower and upper respectively.

3-58. Data Words.

Data Word A - Card 5 - latitude (20 bits)

Data Word B - Card 6 - longitude (20 bits)

Data Word C - Card 7 - Sine and cosine of magnetic variation. One sign bit (either 1 or 0) and nine magnitude bits each.

Data Word D - Card 8 - Type and elevation. The type is formed according to table 3-10.

Table 3-10. Data Word D Type Code 5

Type	ILS	DME	VOR	UHF-DF	TAC BRG	
LVR	0	0	01	0	0	0
MVR	0	0	10	0	0	0
HVR	0	0	11	0	0	0
LVR DME	0	1	01	0	0	0
MVR DME	0	1	10	0	0	0
HVR DME	0	1	11	0	0	0
LVR LTC	0	1	01	0	0	1
LVR MTC	0	1	01	0	1	0
LVR HTC	0	1	01	0	1	1
MVR LTC	0	1	10	0	0	1
MVR MTC	0	1	10	0	1	0
MVR HTC	0	1	10	0	1	1
HVR LTC	0	1	11	0	0	1
HVR MTC	0	1	11	0	1	0
HVR HTC	0	1	11	0	1	1
LTC	0	1	00	0	0	1
MTC	0	1	00	0	1	0
HTC	0	1	00	0	1	1
UHF-DF	0	0	00	1	0	0

3-59. Upon completion of the Radio Facility Data Sheet, data preselector information regarding each radio station is manually precoded into eight or sixteen 20-bit binary data words. Each word is written on the keyboard as eight octal digits. An example of a data preselector word and binary equivalent is illustrated in figure 3-10, where dn is the octal data bits, 0 are bits not recognized and X are valid binary bits.

d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
0XX	XXX	0XX	XXX	XXX	XXX	0XX	0XX

Figure 3-10. Data Preselector Word

3-60. Each data preselector word occupies three lines of punched paper tape and is preceded by a dummy code. Figure 3-11 illustrates the tape format for a data preselector word.

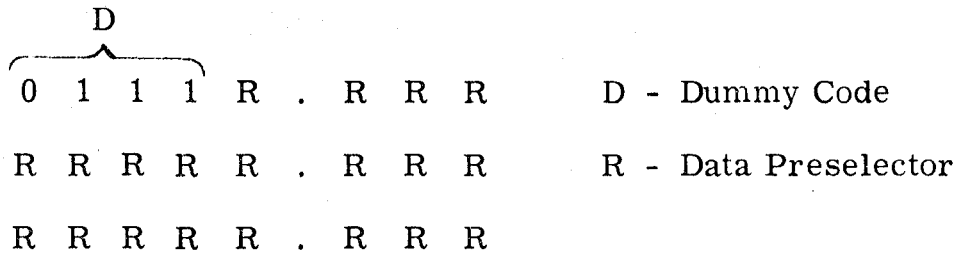


Figure 3-11. Tape Format, Data Preselector Word

3-61. A sample "Radio Facilities Data" sheet for a type 5 is illustrated in figure 3-12 Part (A). Figure 3-12 Part (B) is the information contained in figure 3-12 Part (A) as it would appear on paper tape.

3-62. Figure 3-13 shows the band location in octal of the various radio facilities simulated in the Mark I. (Marker call letters are used for reference only.) The symbol J in figure 3-13 and table 3-11 represents the serial number of the individual station within a group. The J numbers range from 0 to 37, thus the information for the twelfth station is obtained by assigning the numerical value 11 to the algebraic symbol J.

3-63. Table 3-11 is used in conjunction with figure 3-13 to find the location of any part of the facilities being simulated. For example: the sine and cosine axis for the middle marker station located at Kansas City, Mo. required a change. Referring to paragraph 3-26 it is found that the sign and cosine axis is assigned to data word "C" for Middle and Outer Markers. The formula for locating data word "C" is Base Address +J +100. The data word base address for middle markers is 0200 (figure 3-13) and the J number for Kansas City is 07. The sine and cosine axis can therefore be found at drum location 0307 octal or 0199 decimal.

3-64. Figure 3-14 illustrates the "layout" of the information on the data preselector band. All numerics in this figure are in octal form.

Table 3-11. Data Preselector Word Storage Locations

Word	Card Number	Octal Address*
1 (Preselector Word 1)	1	Base Address +4J
2 (Preselector Word 2)	2	Base Address +4J +1
3 (Preselector Word 3)	3	Base Address +4J +2
4 (Preselector Word 4)	4	Base Address +4J +3
5 (Data Word A)	5	Base Address +J

Table 3-11. Data Preselector Word Storage Locations (Cont)

Word	Card Number	Octal Address*
6 (Data Word B)	6	Base Address +J +40
7 (Data Word C)	7	Base Address +J +100
8 (Data Word D)	8	Base Address +J +140
9 (First Call Letter Group)	9	Base Address +J +200
10 (Second Call Letter Group)	10	Base Address +J +240
11 (Third Call Letter Group)	11	Base Address +J +300
12 (Fourth Call Letter Group)	12	Base Address +J +340
13 (Data Word D)	13	Base Address +J +400
14 (Data Word F)	14	Base Address +J +440
15 (Data Word G)	15	Base Address +J +500
16 (Data Word H)	16	Base Address +J +540

*All arithmetic performed under this heading is octal arithmetic.

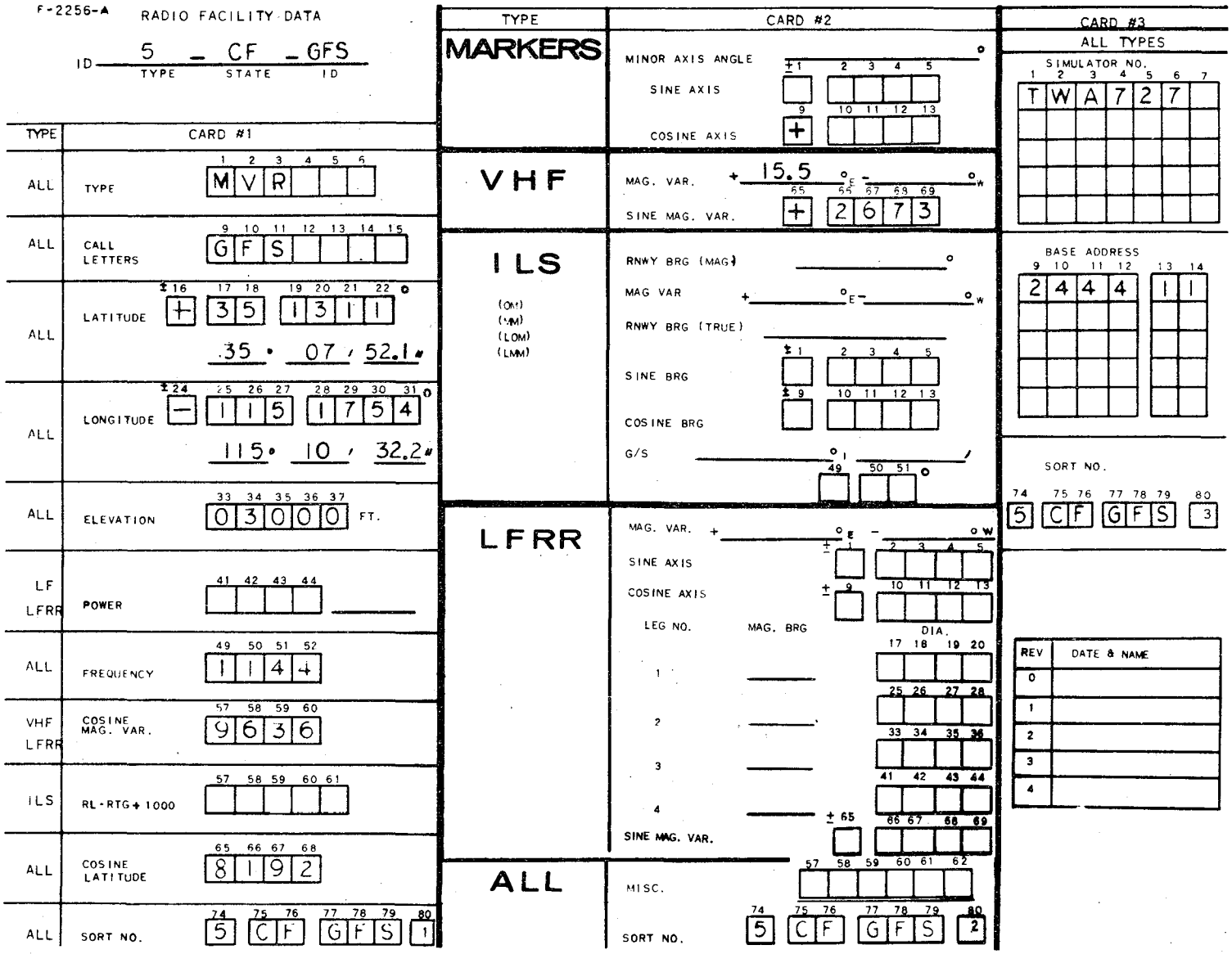
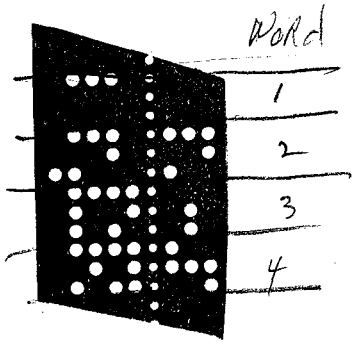


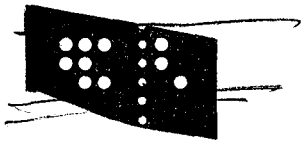
Figure 3-12. Data Preselector Program Part A - Radio Facility Data Sheet

MARK I

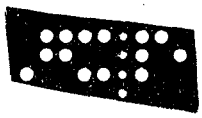


PRESELECT WORDS
WORD 1 - ZERO
WORD 2 - FREQUENCY
WORD 3 - LATITUDE (U AND L)
WORD 4 - LONGITUDE (U AND L)

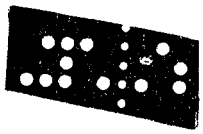
DATA WORDS



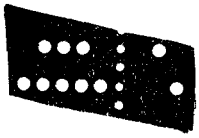
A - LATITUDE



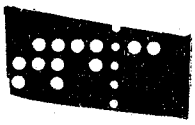
B - LONGITUDE



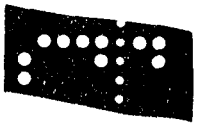
C - SIN AND COSINE (MAG. VAR.)



D - TYPE AND ELEV.

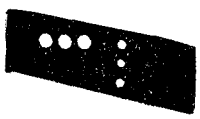


CALL LETTERS - 1



- 2

- 3



- 4

Figure 3-12. Data Preselector Program Part B - Punched Paper Tape

Preselector Words Base Address →	0000		0400		1000		1600		2400		3200		4000		4600		5400		6200		7000	
J / 4J	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters	Type	Call Letters
0 / 0					F	11.3			H	TBC	HV	PGS	MV	RBV					-H-	LAX		
1 / 4	MM	LAX	OM	LAX	B	31	I	LAX	H	ONT	MV	LGB	LV	STW	LO	LA	LM	AX	-H-	DOW		
2 / 10	MM	SFO	OM	SFO	B	13.1	I	SFO	L	SMO	LV	VTU	MV	SAX	LO	SF	LM	FO	MH	FRX		
3 / 14	MM	MFA	OM	MFA			I	MFA	L	LHS	MV	FIM	MV	SBJ	LO	MF	LM	FA	MH	HWD		
4 / 20	MM	MIA	OM	MIA			I	MIA	H	PMD	MV	GMN	HV	FMN	LO	MI	LM	IA	MH	SFG		
5 / 24	MM	ORD	OM	ORD			I	ORD	L	SXC	HV	SBA	HV	IDL	LO	OR	LM	RD	-H-	MIA		
6 / 30	MM	IAC	OM	IAC			I	IAC	M	PYE	HV	SAC	MV	RVH	LO	IA	LM	AC	MH	TMT		
7 / 34	MM	MKC	OM	MKC			I	MKC	L	SFO	HV	SCK	MV	HUO	LO	MK	LM	KC	-H-	PRR		
10 / 40	MM	MCI	OM	MCI			I	MCI	L	SJC	LV	SAU	MV	DAY	LO	MC	LM	CI	MH	NBU		
11 / 44	MM	STJ	OM	STJ			I	STJ	M	GFS	HV	OAK	MV	CLE	LO	ST	LM	TJ	MH	PLV		
12 / 50	MM	EWR	OM	EWR			I	EWR	L	DCA	HV	HEC	MV	TVE	LO	EW	LM	WR	MH	FRY		
13 / 54	MM	LGA	OM	LGA			I	LGA	L	FLL	MV	DAG	MV	ABE	LO	LG	LM	GA	MH	LIY		
14 / 60	MM	IDL	OM	IDL			I	IDL	M	BSY	HV	ALS	HV	PIT	MH	ID	LM	DL	MW	C		
15 / 64	MM	IWY	OM	IWY			I	IWY	M	API	MV	PUB	HV	ERI	LO	IW	LM	WY	MH	SRI		
16 / 70	MM	DLX	OM	DLX			I	DLX	L	DPA	HV	MIA	MV	PSB			LM	LX				
17 / 74	MM	DIA	OM	DIA			I	DIA	M	BDF	MV	JOT	MV	LDN	LO	DI	LM	IA				
20 / 100									H	HLC	HV	OBK	LV	CSN								
21 / 104									M	ICT	MV	CGT	MV	HRN								
22 / 110									L	FDK	LV	RFD	MV	MRB								
23 / 114									L	EMI	MV	EON	LD	LAX								
24 / 120									L	RIS	MV	MLI	MV*	APC								
25 / 124									H	BLD	HV	DSM	LD	OSI								
26 / 130									L	BLM	MV	TOP	LD	ORD								
27 / 134									M	TRN	HV	GCK	LD	COL								
30 / 140									L	LGA	MV	SLN										
31 / 144									L	DPK	HV	BAL										
32 / 150									M	ARD	HV	PMM										
33 / 154									M	FRR	LV	STJ										
34 / 160									H	OAF	HV	MKC										
35 / 164									M	BSP	HV	STL										
36 / 170											MV	PWE										
37 / 174											MV	CYN										
Data Words → Base Address	MM 0200		OM 0600		FAN/Z 1200		VHF 1 2000		VHF 2 2600		VHF 3 3400		VHF 4 4200		LF 1 5000		LF 2 5600		LF 3 6400		LF 4 7200	
	MARKERS						VHF/UHF								LOW FREQUENCY							

NOTES:

*USED AS MVORDME

TYPE CODE:
B = BONE
F = FM
-H- = H
H = HVR
HV = HVORTAC
I = ILS
L = LVR
LD = LVORDME
LM = LMM
LO = LOM
LV = LVORTAC
M = MVR
MH = MH
MM = MM
MV = MVORTAC
MW = MHW
OM = OM

Figure 3-13. Radio Facilities Band Locations (Octal)

	DRUM WORDS	FIELD 1 MIDDLE MARKER 0000-0377	FIELD 2 OUTER MARKER 0400-0777	FIELD 3 FAN/Z MARKER 1000-1577	FIELD 4 VHF GROUP 1 1600-2377	FIELD 4 VHF GROUP 2 2400-3177	FIELD 4 VHF GROUP 3 3200-3777	FIELD 4 VHF GROUP 4 4000-4577	FIELD 5 LF GROUP 1 4600-5377	FIELD 5 LF GROUP 2 5400-6177	FIELD 5 LF GROUP 3 6200-6777	LF/AN GROUP 4 7000-7777
PRESELECTION CONTROL DATA (NOT TRANSFERABLE)	BLANK	0 -	400 -	1000 -	1600 -	2400 -	3200 -	4000 -	4600 -	5400 -	6200 -	7000 -
	fL fU	1 STA 0	401 STA 0	1001 STA 0	1601 STA 0	2401 STA 0	3201 STA 0	4001 STA 0	4601 STA 0	5401 STA 0	6201 STA 0	7001 STA 0
	Xl Xu	2 STA 0	402 STA 0									
	Yl Yu	3 STA 0	403 STA 0									
	BLANK	4 -	404 -	1004 -								7004 -
	fL fU	5 STA 1	405 STA 1									
	Xl Xu	6 STA 1	406 STA 1									
	Yl Yu	7 STA 1	407 STA 1									
	BLANK	10 -	410 -	1010 -								7010 -
	fL fU	11 STA 2	411 STA 2									
	Xl Xu	12 STA 2	412 STA 2				STATION PATTERN IDENTICAL FOR ALL FIELDS					
	Yl Yu	13 STA 2	413 STA 2									
		↓	↓									
	BLANK	174 -	574 -	1174 -	1774 -	2574 -	3374 -	4174 -	4774 -	5574 -	6374 -	7174 -
	fL fU	175 STA 37	575 STA 37									
	Xl Xu	176 STA 37	576 STA 37									
	Yl Yu	177 STA 37	577 STA 37									
INFORMATION FOR TRANSFER TO CORE MEMORY	DATA A	200 STA 0 201 STA 1 202 STA 2	600 STA 0 601 STA 1 602 STA 2	1200 STA 0	2000 BLANK 2001 STA 1	2600 STA 0	3400 STA 0	4200 STA 0	5000 STA 0	5600 STA 0	6400 STA 0	7200 STA 0
		↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	DATA A	237 STA 37	637 STA 37	1237 STA 37	2037 STA 37	2637 STA 37	3437 STA 37	4237 STA 37	5037 STA 37	5637 STA 37	6437 STA 37	7237 STA 37
	DATA B	240 STA 0 241 STA 1	640 STA 0 641 STA 1	1240 STA 0	2040 STA 0	2640 STA 0	3440 STA 0	4240 STA 0	5040 STA 0	5640 STA 0	6440 STA 0	7240 STA 0
		↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	DATA B	277 STA 37	677 STA 37	1277 STA 37	2077 STA 37	2677 STA 37	3477 STA 37	4277 STA 37	5077 STA 37	5677 STA 37	6477 STA 37	7277 STA 37
	DATA C	300 STA 0	700 STA 0	1300 STA 0	2100 STA 0	2700 STA 0	3500 STA 0	4300 STA 0	5100 STA 0	5700 STA 0	6500 STA 0	7300 STA 0
		↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	DATA C	337 STA 37	737 STA 37	1337 STA 37	2137 STA 37	2737 STA 37	3537 STA 37	4337 STA 37	5137 STA 37	5737 STA 37	6537 STA 37	7337 STA 37
	DATA D	340 STA 0	740 STA 0	1340 STA 0	2140 STA 0	2740 STA 0	3540 STA 0	4340 STA 0	5140 STA 0	5740 STA 0	6540 STA 0	7340 STA 0
		↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	DATA D	377 STA 37	777 STA 37	1377 STA 37	2177 STA 37	2777 STA 37	3577 STA 37	4377 STA 37	5177 STA 37	5777 STA 37	6577 STA 37	7377 STA 37
	FIRST CALL LETTER GROUP			1400 STA 0 ↓ 1437 STA 37	2200 STA 0 ↓ 2237 STA 37	3000 STA 0 ↓ 3037 STA 37	3600 STA 0 ↓ 3637 STA 37	4400 STA 0 ↓ 4437 STA 37	5200 STA 0 ↓ 5237 STA 37	6000 BLANK ↓ 6037 STA 37	6600 STA 0 ↓ 6637 STA 37	7400 STA 0 ↓ 7437 STA 37
	SECOND CALL LETTER GROUP			1440 STA 0 ↓ 1477 STA 37	2240 STA 0 ↓ 2277 STA 37	3040 STA 0 ↓ 3077 STA 37	3640 STA 0 ↓ 3677 STA 37	4440 STA 0 ↓ 4477 STA 37	5240 STA 0 ↓ 5277 STA 37	6040 STA 0 ↓ 6077 STA 37	6640 STA 0 ↓ 6677 STA 37	7440 STA 0 ↓ 7477 STA 37
	THIRD CALL LETTER GROUP			1500 STA 0 ↓ 1537 STA 37	2300 STA 0 ↓ 2337 STA 37	3100 STA 0 ↓ 3137 STA 37	3700 STA 0 ↓ 3737 STA 37	4500 STA 0 ↓ 4537 STA 37	5300 STA 0 ↓ 5337 STA 37	6100 STA 0 ↓ 6137 STA 37	6700 STA 0 ↓ 6737 STA 37	7500 STA 0 ↓ 7537 STA 37
	FOURTH CALL LETTER GROUP			1540 STA 0 ↓ 1577 STA 37	2340 STA 0 ↓ 2377 STA 37	3140 STA 0 ↓ 3177 STA 37	3740 STA 0 ↓ 3777 STA 37	4540 STA 0 ↓ 4577 STA 37	5340 STA 0 ↓ 5377 STA 37	6140 STA 0 ↓ 6177 STA 37	6740 STA 0 ↓ 6777 STA 37	7540 STA 0 ↓ 7577 STA 37
	DATA E											7600 STA 0 ↓ 7637 STA 37
	DATA F											7640 STA 0 ↓ 7677 STA 37
	DATA G											7700 STA 0 ↓ 7737 STA 37
	DATA H											7740 STA 0 ↓ 7777 STA 37

NOTES:

1. All marker stations operate on 75 MCS. - Preselection by X/Y only.
2. X/Y position required for preselection during interval 2-7177. Core memory access 7200 0000.
3. VHF receiver frequency required for preselection during interval 1601-4175. Core memory access 0001 1600.
4. LF receiver frequency required for preselection during interval 4577-7170. Core memory access 4200 4600.
5. Preselection patterns:

Drum Rev #1	#2	#3	#4
Middle Mk.	Middle Mk.	Middle Mk.	Middle Mk.
Outer Mk.	Outer Mk.	Outer Mk.	Outer Mk.
Fan/Z Mk.	Fan/Z Mk.	Fan/Z Mk.	Fan/Z Mk.
VHF Rcvr. #1	VHF Rcvr. #2	VHF Rcvr. #3	VHF Rcvr. #4
LF Rcvr. #1	LF Rcvr. #2	LF Rcvr. #1	LF Rcvr. #2

6. Two stations are deleted due to blanking requirements during drum loading. These blank words have all bits equal to zero and are non-recoverable words. The stations deleted are:
   a. Field 4, VHF Group 1, Station #1
   b. Field 5, LF Group 2, Station #1
7. Number of stations available for preselection:
   32 Middle Marker
   32 Outer Marker
   32 Fan/Z Marker
   127 VHF
   95 ~~LF~~ ADF
   32 ~~LF/AN~~
   350 Total

Figure 3-14. Data Preselector Band

3-65. MULTIPLEXER PROGRAM.

3-66. The 100 input lines of the A/D converter are programmed sequentially on the four slow bands (approximately 25 inputs to a band). This input information (analog voltages) is transferred sequentially from the Multiplexer to the Multiverter, (where they are converted to digital), and further transferrred to a main core memory location and on to another core location depending upon the relay positions in the Redcor Multiplexer. (See figure 3-15.)

3-67. A Boolean program used in conjunction with the last multiplexed program on slow band four is used to simulate resetting the switch positions to 0. Figure 3-16 is a sample multiplexer program for slow band 4.

3-68. The three other multiplexer programs (slow bands 1, 2, and 3) are ended with a small Boolean program consisting of BLD, BIN, and BST into the same core location as the BLD instruction. This is done to change switch position (A to $\overline{A}$, B to $\overline{B}$ etc.).

3-69. BOOLEAN EQUATIONS.

3-70. General. There are five basic steps which should be taken into consideration by the programmer prior to programming a Boolean equation. These steps will help the programmer to visualize the system more clearly and eliminate possible extra program steps. The five basic steps are as follows:

a. Develop a complete "picture" of the system.

(1) Block diagram or signal flow type.

(2) Show all indicators, lights, etc.

(3) Show all actuators, circuit breakers, etc.

(4) Determine the extent of simulation.

b. Redraw the system "picture" including the results of step 4 of a.

c. Determine any breakdown of the system.

(1) Any indicator, such as a light, must be a Boolean output. Any breakdown should be in these terms.

(2) Any term that is used many times should be stored in memory.

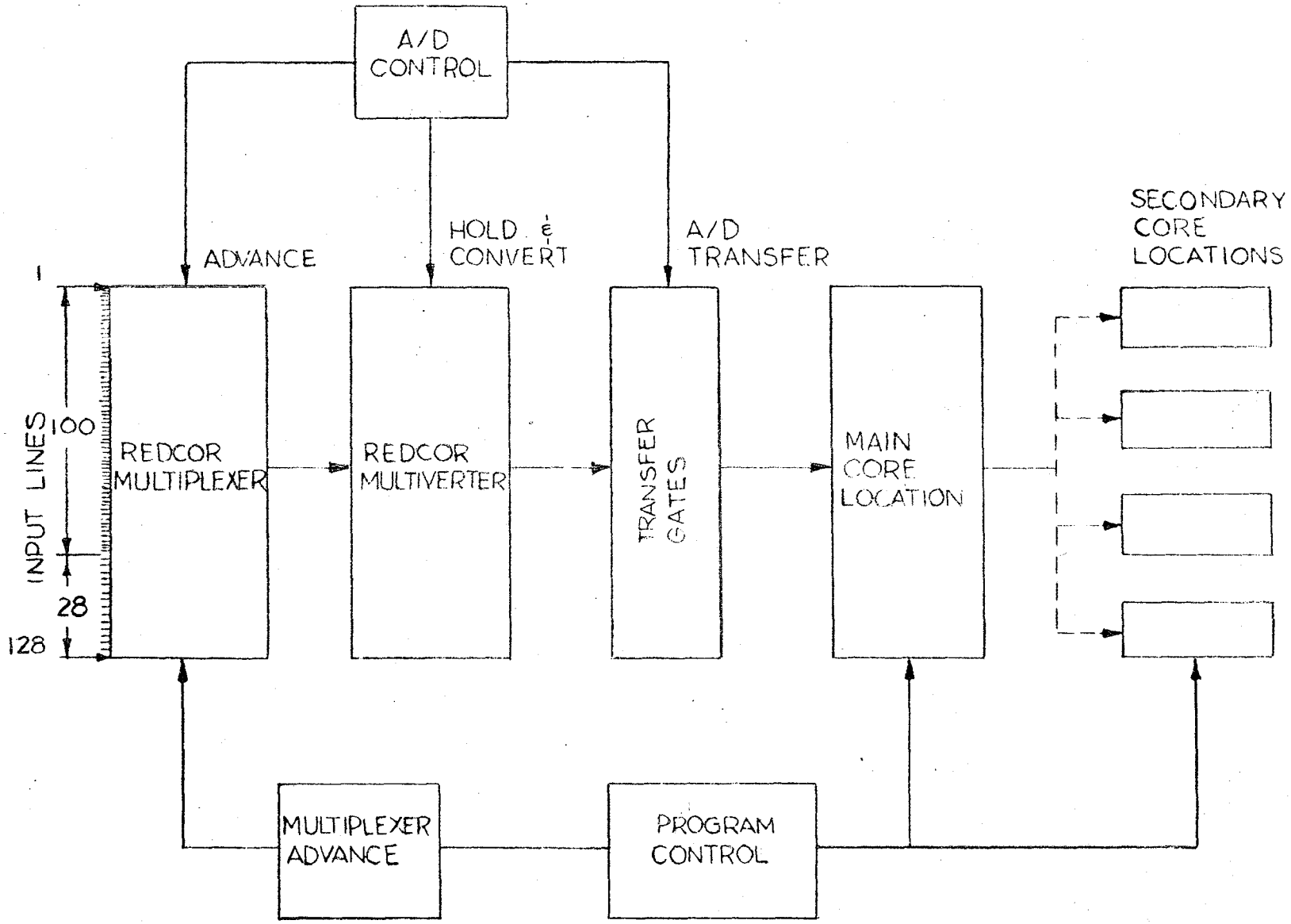


Figure 3-15. Block Diagram, Multiplexer Flow of Information

DATE 5/9/63

CODING AND CONSTANT SHEET

PROBLEM MULTIPLEXER PROGRAM - SLOW BAND 8 QUADRANT 1

MARK I COMPUTER

PROGRAMMER JOHN DOE

F-2214-A

INSTRUCTION NUMBER (1 2 3 4 5 6 7)	MNEMONIC CODE	O.P. CODE (8 10)	MEM. ADD. (12 13 14 15)	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	LD	20	0270									Steps 1 Thru 32 Used To Load Contents Of Main Core
02	ST	23	2700									Memory Location And Store Those Contents Into A
03	LD	20	0271									Certain Other Core Location Dependent Upon The Relay
04	ST	23	2704									Position In The Packard-Bell Multiplexer.
05	LD	20	0272									
06	ST	23	2710									
07	LD	20	0273									
08	ST	23	2714									
09	LD	20	0274									
10	ST	23	2720									
11	LD	20	0275									
12	ST	23	2724									
13	LD	20	0276									
14	ST	23	2730									
15	LD	20	0277									
16	ST	23	2734									
17	LD	20	0300									
18	ST	23	2740									
19	LD	20	0301									
20	ST	23	2744									
21	LD	20	0302									
22	ST	23	2750									
23	LD	20	0303									
24	ST	23	2754									
25	LD	20	0304									
26	ST	23	2760									
27	LD	20	0305									
28	ST	23	2764									
29	LD	20	0306									
30	ST	23	2770									

Figure 3-16. Multiplexer Program (Sheet 1 of 2)

DATE 5/9/63

PROBLEM MULTIPLEXER PROGRAM - SLOW BAND 8 QUADRANT 1

PROGRAMMER JOHN DOE

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-2214-A

PAGE 2 OF 2

INSTRUCTION NUMBER (1 2 3 4 5 6 7)	MNEMONIC CODE	O.P. CODE (9 10)	MEM. ADD. (12 13 14 15)	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
31	LD	20	0307									
32	ST	23	2774									
33	BLD	33	2405				A or B					
34	BIN	32	0000				$\overline{A}$ or $\overline{B}$					
35	BLD	33	2405				A or B					
36	AND	31	0000				(A or B) ($\overline{A}$ or $\overline{B}$)					Ensure 0
37	BST	34	2405				0					Store Zero In Location A or B
38	BST	34	2406				0					Store Zero In Location A or B

Figure 3-16. Multiplexer Program (Sheet 2 of 2)

d. Write the Boolean expressions for each section of the system breakdown.

(1) Use combinations that are common to several sections of the system and can be reclaimed from the temporary Boolean register.

e. Write the programming instructions.

3-71. In order that the "pictures" of the system transfer the maximum information, a standard method of representation should be used. A variation of the signal flow graph is most easily understood.

3-72. A function is defined by the lines directed into the circle with that label. These lines come from other labeled functions or circles labeled with Boolean functions such as "AND" or "OR". (See figure 3-17.)

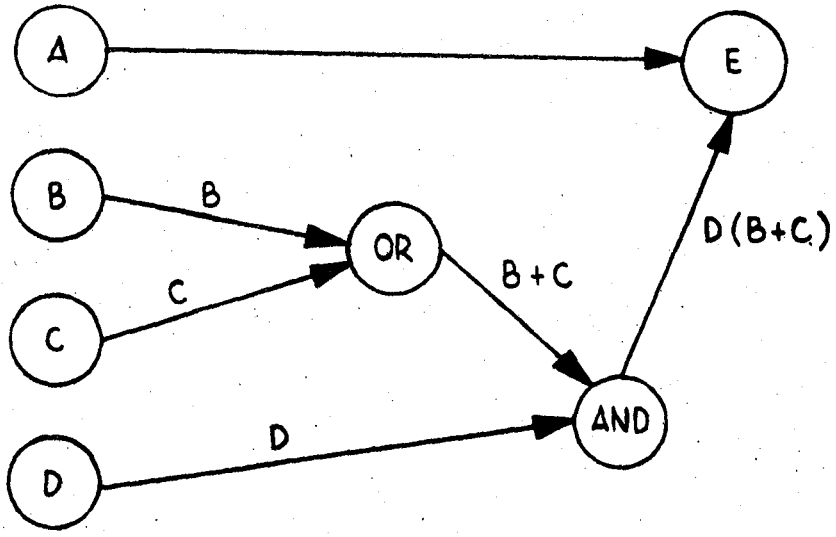


Figure 3-17. Flow Chart

3-73. The function E is defined by the two lines directed into it. One of these is the function A, the other is the indicated function of B, C, and D. The equation for E would be E = A + D (B + C) in Boolean designation.

3-74. The circles can be labeled with the names of switches, lights, circuit breakers, relays, and other elements of the aircraft's system. Using this method the systems as shown in the aircraft manufacturer's manuals can be transcribed to a function diagram. Flow charts are simplified until only the inputs, outputs, and variables which are to be stored remain as labeled circles.

3-75. The Boolean expressions for the function represented by a labeled circle can be written directly from the flow chart. The function of any labeled circle is the sum of the directed lines entering the circle. Any directed line leaving an "OR" circle is the sum of the directed lines entering the circle. Any directed

line leaving an "AND" circle is the product of the directed lines entering the circle. Utilization of these ideas result in the Boolean function for all outputs and elements to be stored.

3-76. An illustration of this method is shown in figures 3-18 and 3-19. Figures 3-18 is a typical aircraft power circuit. All details except those necessary to determine when the three buses are energized have been omitted. Figure 3-19 represents the transition from figure 3-18 to determine the equations for energization of the three buses. The circles A and B on figures 3-19 represent Gen I operating and Gen I circuit breaker "made". To the right of Gen I C.B. on figure 3-18 there is a junction corresponding to the "AND" circle (figure 3-19) which the arrows from A and B enter. This junction is defined by AB and each arrow leaving the circle is so designated. In the circuit, power from this junction goes through GEN I RY to a second junction. This junction is represented on figure 3-19 by the "AND" circle which the arrows A, B, and C enter. C is the circle for GEN I RY. Power from this junction feeds LOAD BUS #1, but this bus can also be fed through the GEN I BUS TIE. Therefore, the arrow ABC must enter an "OR" circle which then goes to E (LOAD BUS #1). The first junction also supplies power to the ESS PWR BUS through the GEN I ESS PWR relay. This is shown by the arrow AB going to the upper right hand "AND" circle in conjunction with the D arrow. This "AND" circle goes to G through an "OR" circle indicating two power sources for the ESS PWR BUS "G". The rest of the circuit of figure 3-18 is described on figure 3-19 in a similar manner.

3-77. The Boolean equations for G, E, and K can be determined from the arrows entering the "OR" circles which define G, E, and K.

$$G = ABD + LMN$$

$$E = ABC + FHJLM$$

$$K = JLM + FHABC$$

3-78. When the Boolean equations have been developed, a program can be written. This program is in terms of storage, input, and output locations in the Mark I. On a system design sheet such as figure 3-19, a tabulation of the letter designations used on that sheet, their corresponding aircraft functions and Mark I storage locations should be included. This tabulation provides the correlation between the Mark I locations and the simulated aircraft variables. This tabulation would appear as follows:

GEN I OPER	-A-	B2347
GEN I C.B.	-B-	B1433
GEN I RY	-C-	B2131
GEN I ESS PWR RY	-D-	B2134
LOAD BUS 1	-E-	B3214
GEN I BUS TIE RY	-F-	B2133
ESS PWR BUS	-G-	B3217
GEN II BUS TIE RY	-H-	B2132
GEN II RY	-J-	B2135
LOAD BUS 2	-K-	B3215
GEN II C.B.	-L-	B1435
GEN II OPER	-M-	B2354
GEN II ESS PWR RY	-N-	B2136

3-79. A preliminary program sheet can be filled out to ensure the programmer that all phases of the problem are included in the program. This step further eliminates the possibility of errors, or insufficient information to solve a problem being programmed into Mark I. A program sheet of this type developed for figure 3-19 follows:

	ACCUM.	SALVAGE REGISTER			
BLD - A	A				
AND - B	AB				
BLD - D	D	AB			
AND - 0000	ABD	AB			
BLD - L	L	ABD	AB		
AND - M	LM	ABD	AB		
BLD - N	N	LM	ABD	AB	
AND - 0000	LMN	LM	ABD	AB	
OR - 0001	ABD + LMN = G	LM	ABD	AB	
BST	G	LM	ABD	AB	
BLD - C	C	G	LM	ABD	AB
AND - 0003	ABC	G	LM	ABD	AB
BLD - J	J	ABC	G	LM	ABD
AND - 0002	JLM	ABC	G	LM	ABD
BLD - H	H	JLM	ABC	G	LM
AND - F	HF	JLM	ABC	G	LM
AND - 0000	HFJLM	JLM	ABC	G	LM
OR - 0001	ABC + FHJLM = E	JLM	ABC	G	LM

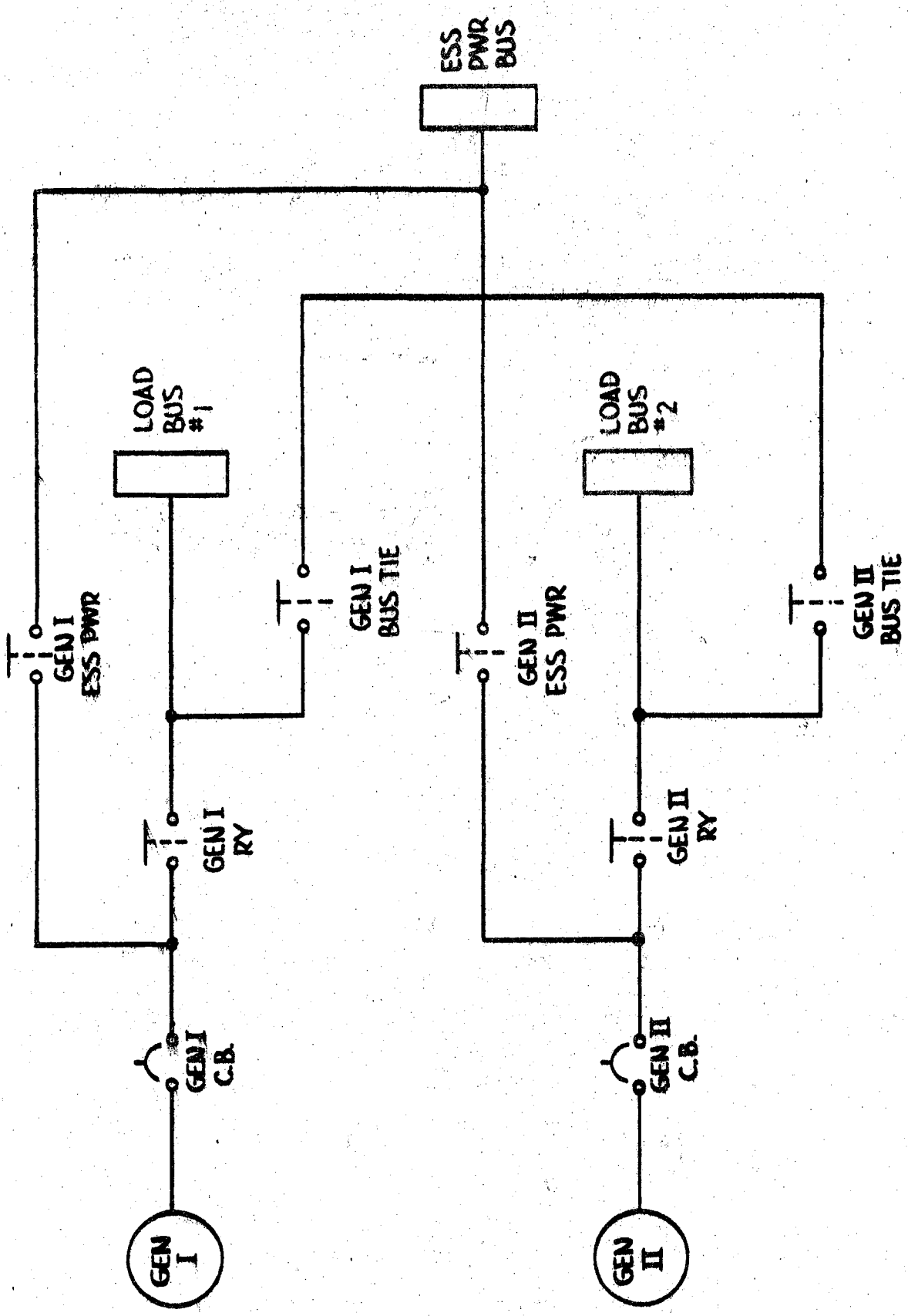


Figure 3-18. Aircraft Power Circuit

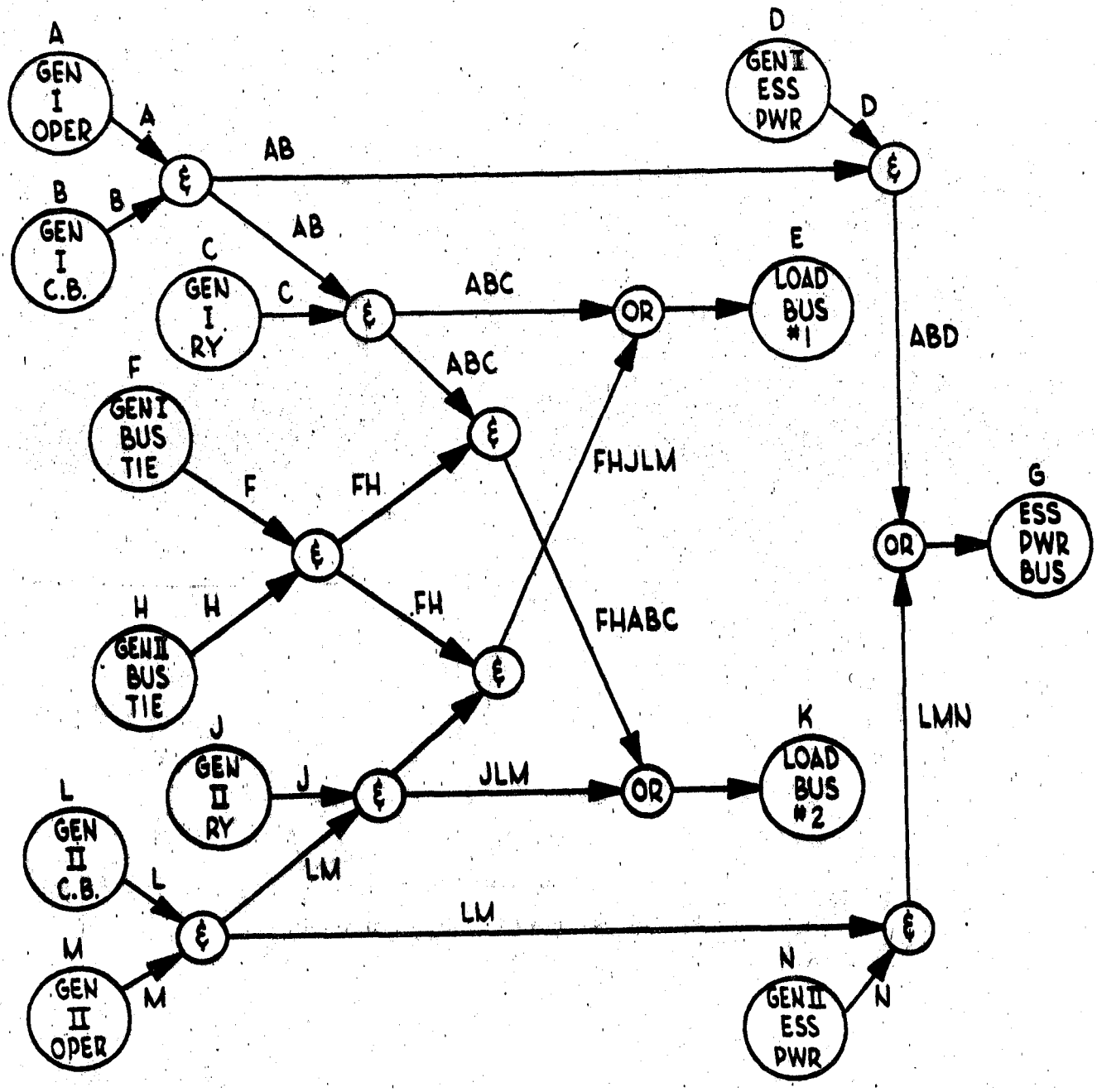


Figure 3-19. Aircraft Power Circuit Flow Chart

DATE 5/7/63

PROBLEM BUS ENERGIZATION

PROGRAMMER H. SMITH

F-2214-A

CODING AND CONSTANT SHEET

MARK I COMPUTER

PAGE 1 OF 1

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	BLD	33	2347				A					System Diagram (Figure 3-12)
02	AND	31	1433				AB					Calculate G
03	BLD	33	2134				D					
04	AND	31	0000				ABD					
05	BLD	33	1435				L					
06	AND	31	2354				LM					
07	BLD	33	2136				N					
08	AND	31	0000				LMN					
09	OR	30	0001				ABD + LMN = G					
10	BST	34	3217				G					
11	BLD	33	2131				C					Calculate E
12	AND	31	0003				ABC					lost
13	BLD	33	2135				J					
14	AND	31	0002				JLM					
15	BLD	33	2132				H					lost
16	AND	31	2133				FH					
17	AND	31	0000				FHJLM					
18	OR	30	0001				ABC + FHJLM = E					
19	BST	34	3214				E					lost
20	BLD	33	2132				H					Calculate K
21	AND	31	2133				FH					
22	AND	31	0002				ABCFH					
23	OR	30	0001				ABCFH + JLM = K					
24	BST	34	3215				K					

Figure 3-20. Bus Energization

	ACCUM.	BOOLEAN SALVAGE			
BST - E	E	JLM	ABC	G	LM
BLD - H	H	E	JLM	ABC	G
AND - F	FH	E	JLM	ABC	G
AND - 0002	FHABC	E	JLM	ABC	G
OR - 0001	JLM + FHABC = K	E	JLM	ABC	G
BST - K	K	E	JLM	ABC	G

3-80. Figure 3-20 is a sample of this program as it would appear on the standard form. The letter designations on the flow chart (figure 3-19) are used on this sheet with a cross reference to the flow chart under "REMARKS". The program is arranged using combinations in the Boolean salvage to reduce the number of instructions.

3-81. CORE MEMORY.

3-82. Core Memory Load Data. Information to be placed in core memory will be written, Most Significant digit first, as nine Octal digits, for the loading of 24-bit binary words. Words will be written in core memory address order. An example of a core memory word and binary equivalent is shown in figure 3-21.

Where dn = Octal data bits and sign
X = Valid binary bits, "0" is not recognized
Sign *X = "0" for positive number, "1" for negative number

d_1 (sign)	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
00X	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XX0

Figure 3-21. Core Memory Word

3-83. Core Memory Tape Format. The form of the Core Memory tape data words is shown in figure 3-22 where M = memory data and D = dummy code.

D

0 1 1 1 M . M M M
M M M M M . M M M
M M M M M . M M M
M M M M 0 . 0 0 0

Figure 3-22. Core Memory Word Tape Format

3-84. Figure 3-23 illustrates three different core memory tapes used in testing the core. "A" tape is used for loading all ones, "B" all zeros, and "C" a combination of zeros and ones.

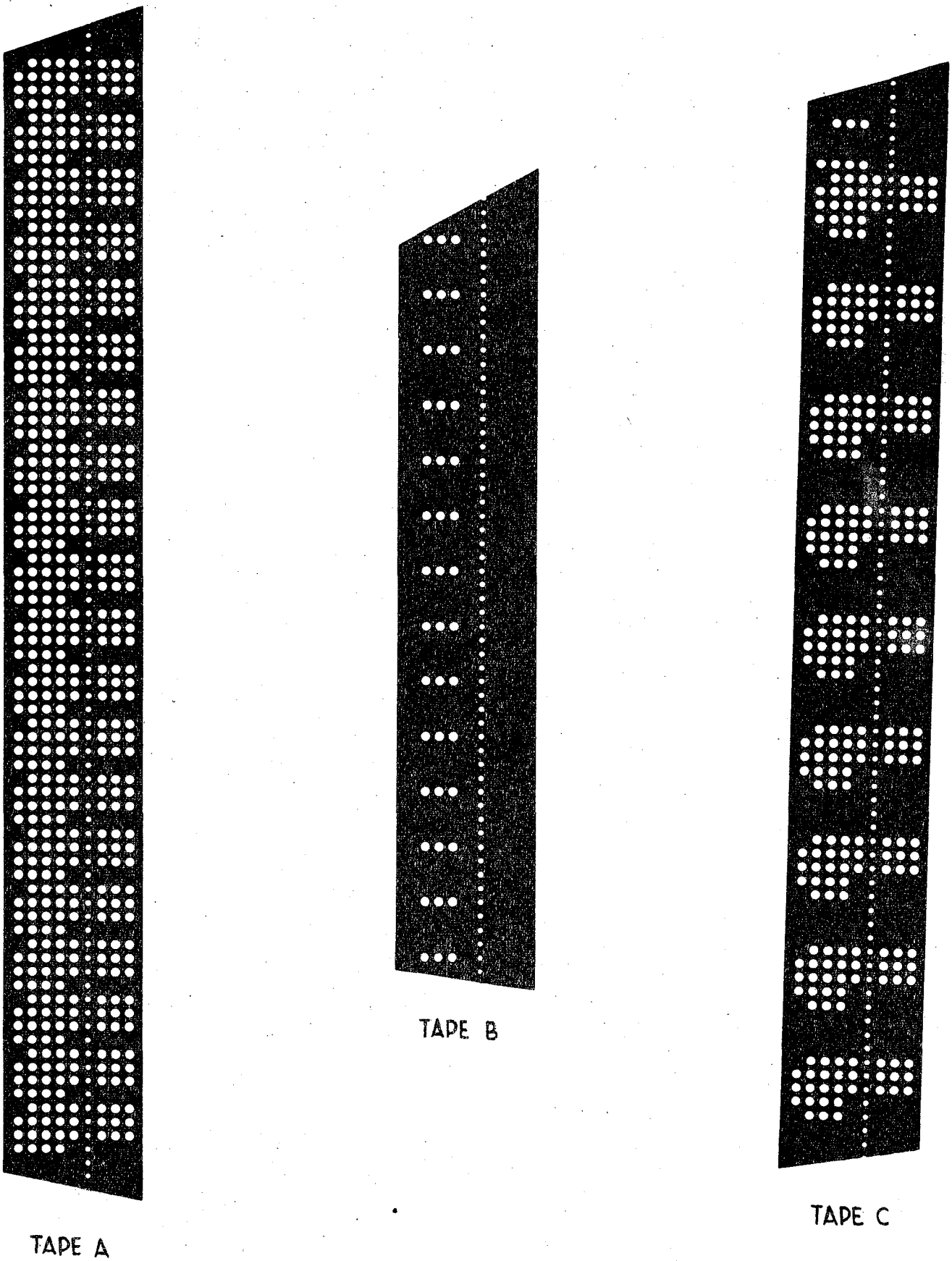


Figure 3-23. Core Memory Tapes

SECTION IV

PROGRAMMING AIDS

4-1. BOOLEAN ALGEBRA.

4-2. General. Boolean algebra is particularly suited for describing situations where variables can assume only two values. These two values have many designations: on-off, yes-no, plus-minus, one-zero, etc. Variables are designated by letters, and the two possible conditions are shown by the absence or presence of a bar over the letter. To illustrate, A and $\overline{A}$ designate the two possible conditions of the variable A. They are read as "A" and "not A". The bar usually signifies the minor or negative condition. The barred function is known as the complement to its corresponding unbarred function.

4-3. To describe systems of more than one variable, several conventions must be accepted. Figure 4-1 illustrates the "AND" and "OR" circuits common to Boolean algebra.

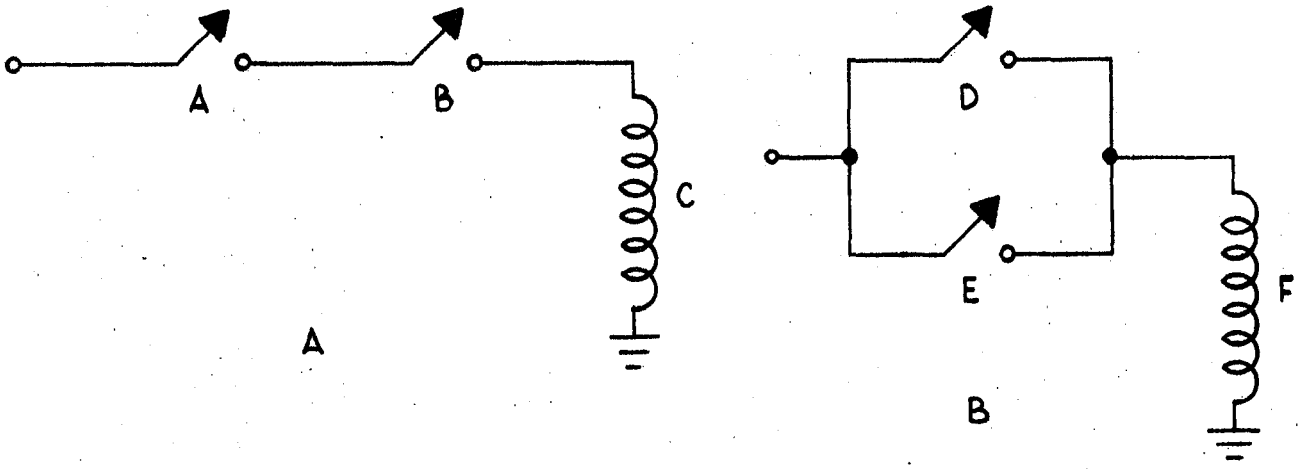


Figure 4-1. "AND" and "OR" Circuits

4-4. In figure 4-1 (A) relay C is energized when switches A and B are both energized. Thus, C equals A and B. The conventional symbol for the "AND" is a multiplication sign or implication ($C = A \cdot B$ or $C = AB$). In (B), the relay F is energized if switch D or E is closed. The symbol for "OR" is the addition sign ($F = D + E$).

4-5. Venn Diagram. A more general illustration of the meaning of "AND" and "OR" is shown by the Venn diagram figure 4-2.

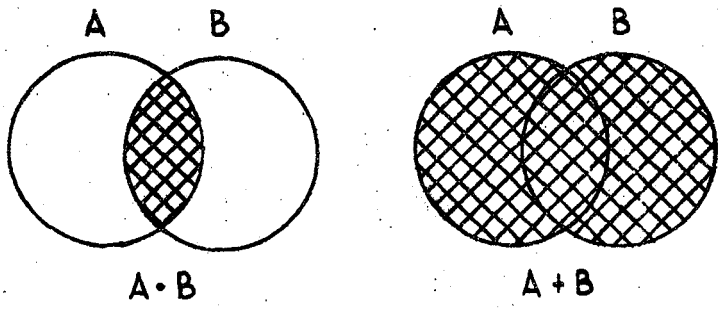


Figure 4-2. Venn Diagram

4-6. Referring to the Venn diagram, figure 4-2, the area inclosed by any circle represents the condition ascribed to the designation for that circle. The cross-hatched area common to both A and B shows the meaning of A·B. The cross-hatched area covering both A and B, shows the meaning of A + B.

4-7. Figure 4-2 also shows the meaning of the complement when applied to more than one variable. Using the circles describing A·B, $\overline{A \cdot B}$ would be all the area not cross-hatched including the area outside the circles. This is the area which is either outside A or outside B. This would be written $\overline{A} + \overline{B}$. This is one of many useful identities. Considering the area describing A + B, $\overline{A + B}$ would be all the area not cross-hatched. It is the area outside of A which is also outside of B. This can be expressed as $\overline{A} \cdot \overline{B}$ giving us the identity $\overline{A + B} = \overline{A} \cdot \overline{B}$.

4-8. Identities. The following are a list of identities useful in manipulating Boolean algebra.

1. $A + 1 = 1$
2. $A \cdot 1 = A$
3. $A + 0 = A$
4. $A \cdot 0 = 0$
5. $A + A = A$
6. $A \cdot A = A$
7. $A + AB = A$
8. $A (A + B) = A$
9. $A + \overline{A} = 1$
10. $\overline{(\overline{A})} = A$
11. $\overline{A + B} = \overline{A} \cdot \overline{B}$
12. $\overline{A \cdot B} = \overline{A} + \overline{B}$
13. $A + \overline{A}B = A + B$
14. $A (\overline{A} + B) = AB$
15. $(A + B) (\overline{A} + C) = AC + \overline{A}B$
16. $(AC + B\overline{C}) = \overline{A}C + \overline{B}\overline{C}$
17. $(A + C) (B + \overline{C}) = (\overline{A} + C) (\overline{B} + \overline{C})$

4-9. The one is used to mean the truth or continuance of a function. It can be compared to a short circuit in a system of switches. Zero is the opposite and can be compared to an open circuit. Number one and number four identities are illustrated in figure 4-3. Identities two and three also concur with figure 4-3.

4-10. Figure 4-3 (A) proves the first identity B = A + 1 = 1. The relay B is always energized whether switch A is open or closed. The fourth identity B = A 0 = 0, (B) shows that B will never be energized. Identities seven and eight are shown in figure 4-4.

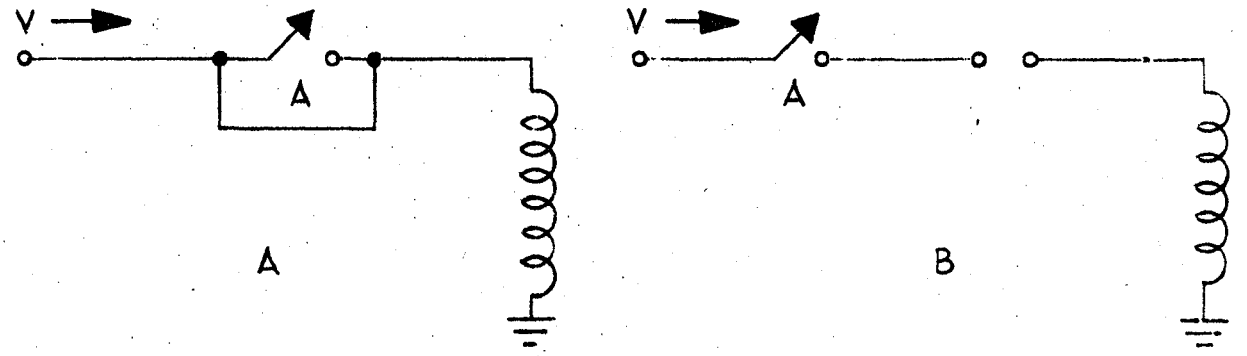


Figure 4-3. Identities 1 through 4

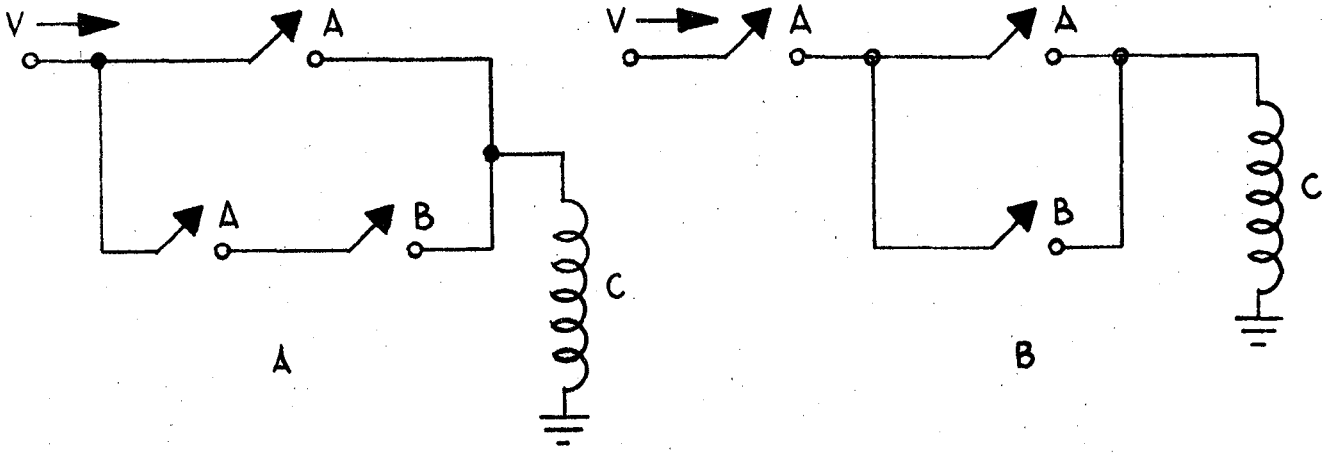


Figure 4-4. Identities Seven and Eight

4-11. Figure 4-4 (A) illustrates the seventh identity, $C = A + AB = A$. The parallel circuit with switches A and B is unnecessary because the operation of the relay C is the same without it. This can be shown mathematically as follows: $C = A + AB = A + A(1 + B) = A + A(1) = A$. The eight identity $C = A(A + B)$ illustrated in (B) shows that the parallel combination of A and B does not change the operation of relay C. This can also be shown by mathematical proof. $C + A(A + B) = A\ \boxed{A(1 + B)} = A\ \boxed{A(1)} = (A) = A$.

4-12. Identity 13, $A + \bar{A}B = A + B$ is derived from the equation $A + AB + \bar{A}B = A + \bar{A}B = A + B$. This equation is illustrated in figure 4-5.

4-13. Switches A and B in the middle path of the circuit (figure 4-5 (A)) do not change the operation of relay C. If either switch A or B is closed there will be a transmission. The mathematical proof follows:

$$A + \bar{A}B = A + AB + \bar{A}B = A + B(A + \bar{A}) = A + B(1) = A + B.$$

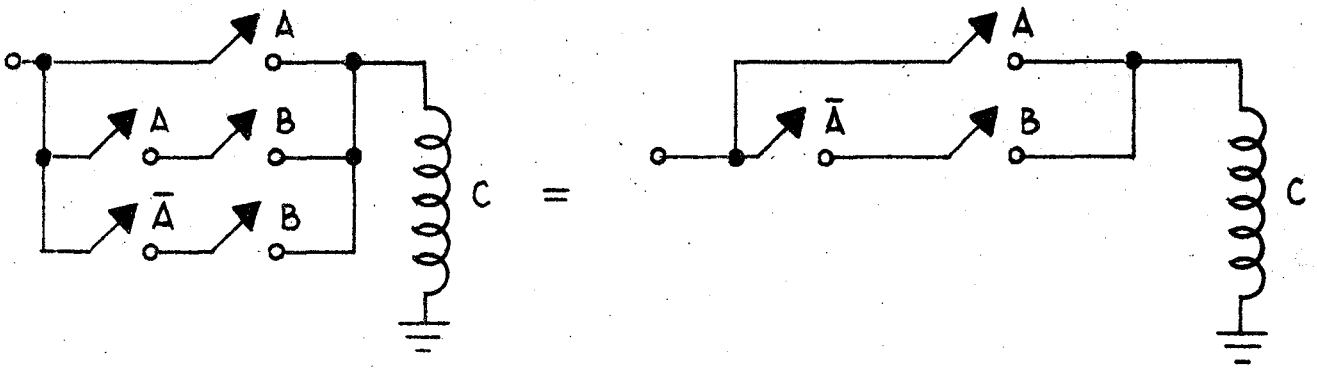


Figure 4-5. Identity 13

4-14. CONDITIONAL SKIPPING.

4-15. General. The "conditional skip" instruction allows the programmer to branch to two different groups of instructions by "skipping over" one of the two groups (i.e., if a certain condition is true, then _____; if not, then _____). The two groups in a branch are listed one after the other in the program. Depending on the results of a test, the Mark I either reads the first group of instructions and skips the second, or it skips the first group and reads the second.

4-16. The state of the Boolean accumulator is the test which determines the branch read. The Mark I will skip only when there is a one in the Boolean accumulator. The programmer is not restricted within the framework of allowed instructions, to devise a test which will result in the Boolean accumulator being set one way or the other so as to reflect the state of the condition being examined.

4-17. Since the two groups of instructions which follow a conditional skip are listed consecutively, then, in the event the first group is not skipped, but is read and performed, the second group must be skipped. In order for the Mark I to skip the second group of instructions, the first group must have a conditional skip as its last instruction. Assuming that the first group of instructions is being read and performed, then the contents of the Boolean accumulator must be zero. It is then necessary to include an "INVERT BOOLEAN" accumulator instruction (this instruction will set the Boolean accumulator to one), prior to the "conditional skip" at the end of the first group. The conditional skip instruction will then cause the second group to be skipped.

4-18. The maximum number of instructions that may be skipped with one "conditional skip" instruction is 127. If there is a necessity for a larger skip, the Mark I can skip from one "conditional skip" instruction to another "conditional skip" instruction.

DATE MAY 6, 1963

PROBLEM X = A + B + C

PROGRAMMER JOHN DOE

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-221a-A

PAGE 1 OF 1

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	LD	20	1234	2^{-7}	A							Load A into the Arithmetic Accumulator
02	SCL	10	1001	2^{-8}	A							Scale Right One Place to Form A.2^{-8}
03	ADD	01	1236	2^{-8}	A + C							Add C.2^{-8} to the Contents of the Accumulator
04	LD	20	1235	2^{-4}	B							Load B into the Accumulator Previous Contents Transfer to Salvage
05	SCL	10	1004	2^{-8}	B							Scale Right Three Places to Form B.2^{-8}
06	ADD	01	0000	2^{-8}	A + B + C							Add Salvage to Accumulator
07	SCL	10	0003	2^{-5}	A + B + C = X							Scale Left Three Places to Form X.2^{-5}
08	ST	23	1237	2^{-5}	X							Store in X Location

MARK I

Figure 4-6. Program Involving Scaling Operation

4-19. SCALING AND SCALING PROBLEMS.

4-20. General. All constants and variables, as handled by the Mark I must be scaled to fall in the range from -1.0 to +1.0. The scaling factor in the Mark I is limited to integral powers of two to minimize time consuming multiplication or division operations. (Scaling or shifting operations are the equivalent of multiplying by powers of two.) The choice of scale factor is limited to the smallest integral power of two which, when divided into X_{max}, results in a number less than or equal to 1.0 (i.e., $X_{max}/2^n \leq 1.0$). For example, some variable that ranged decimally from -80 to +510 would require a scaling factor of at least 2^{-9} or 1/512 to ensure against overflow.

4-21. The Mark I is capable of shifting as many as five places in one basic machine cycle. Thus, a left scale of three places is equivalent to multiplying by 2^{+3}. and right scale of three places is the same as multiplying by 2^{-3}. The scaling or shifting operation requires only one instruction word as opposed to a multiplication which requires five words.

NOTE

Care must be exercised when scaling left to make certain that overflows are not caused by scaling too many places.

4-22. When performing additions and subtractions, the scale factors of all the quantities involved must be identical. If the quantities being added have different scale factors, scaling operations must be performed to make these factors the same. Generally, all of the scales involved must be right scales, eliminating any overflow due to the scaling operation itself. Figure 4-6 illustrates an addition of three variables with different scaling factors.

4-23. Figure 4-6 forms the equation X = A + B + C, with X scaled at 2^{-5} as the desired quantity. A is scaled at 2^{-7}, B at 2^{-4} and C at 2^{-8}. A and B must both be scaled right; one place for A, and four places for B. The terms are then summed to form $X \cdot 2^{-8}$, and this quantity is scaled left three places to form $X \cdot 2^{-5}$.

4-24. When performing an addition and/or subtraction, the risk of overflow is still to be encountered in the actual performance of the operation itself, and serious consideration should be given to the order in which the steps of the equation are listed.

4-25. In the case where A, B, and C are variables ranging from -1.0 to +1.0, rearranging the order of steps is no longer a sufficient guarantee that overflow will not occur. In this case all of the variables must be scaled right a sufficient number of places so that no partial sum can cause an overflow.

4-26. Multiplication. When multiplying two quantities both of which are less than or equal to one, there is no possibility of an overflow. It is possible that the product (AB) might be so small that the number of significant digits remaining in the accumulator are insufficient to describe AB with any accuracy (loss of resolution).

4-27. To overcome loss of resolution it is possible to shift the first fifteen digits of the MQ register into the accumulator using left shift instructions, after a multiplication, thus restoring the resolution of a product. Caution must be taken to ensure that the product AB is small enough to allow the shifting operation without causing overflow.

4-28. Division. When dividing in the Mark I, the problem of overflow is greatly increased. Since the quotient must always be less than or equal to one in magnitude, the dividend must be less or equal to the divisor in magnitude. That is, if $X = \frac{A}{B}$ and $-1.0 \leq X \leq +1.0$ then $/A/ \leq /B/$.

4-29. If A and B are variables, careful consideration must be given to both the scale factors and the behaviors of these variables in order to decide on the final scale factor which will prevent an overflow under any conditions when performing the division.

4-30. A sample problem involving scaling techniques with multiplication and division is illustrated in figure 4-7.

4-31. It is desired to write a program for Mach number. The following equation is to be used: $M = \frac{.0152\ V_p}{\sqrt{T_K}}$

$$T_K = T_{oa} + 273.16$$

where

T_{oa} = temperature of outside air in C^o scaled 2^{-7}
T_K = temperature of outside air in K^o scaled 2^{-9}
V_p = velocity of the A/C along the flight path in ft/sec scaled 2^{-11}
M = Mach number (result) 2^{-1}

Assume that the maximum V_p of the aircraft is 800 mph (1173 ft/sec) and the temperature of the outside air ranges from -70^o to $+70^o$. The first step in programming a problem of this nature is to determine the range of numbers, scaled properly by 2.

$$.3967 \times 2^9 \leq T_K \leq .6702 \times 2^9$$
$$0 \leq V_p \leq .5722 \times 2^{11}$$
$$0 \leq .0152\ V_p \leq .5572 \times 2^5$$
$$.45 \times 2^5 \leq \sqrt{T_K} \leq .58 \times 2^5$$

1 0 0 0 1 0 0 0 0 1 0 0 1

16
256
273.125

DATE MAY 7, 1963
PROBLEM COMPUTE MACH NO.
PROGRAMMER G. GREEN

CODING AND CONSTANT SHEET
MARK I COMPUTER

PAGE 1 OF 1

F-2214-A

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	LD	20	1002	2^{-7}	T_{OA}							
02	SCL	10	1002	2^{-9}	T_{OA}							
03	LDK	22	4211	2^{-9}	273.16							Conversion Factor $C^o \longrightarrow K^o$
04	ADD	01	0000	2^{-9}	$T_{OA} + 273.16 = T_K$							
05	SCL	10	1001	2^{-10}	T_K							
06	SRS	07	1000	2^{-10}	T_K							
07	NPA	12	0000									
08	NPB	25	3777									
09	NPA	12	0000									
10	NPB	25	3777	2^{-5}	$\sqrt{T_K}$							
11	ST	23	1000	2^{-5}	$\sqrt{T_K}$							T_K for Next Iteration
12	LDK	22	7612	2^{+6}	.0152		lost					
13	MLT	03	1004	2^{+6}	.0152							
14	NPA	12	0000									
15	NPB	25	3777									
16	NPA	12	0000									
17	NPB	25	3777	2^{+5}	.0152 V_P							
18	SCL	10	1001	2^{-6}	.0152 V_P							
19	DIV	06	0000	2^{-6}	.0152 V_P							
20	NPA	12	0000									
21	NPB	25	3777									
22	NPA	12	0000									
23	NPB	25	3777	2^{-1}	.0152 $V_P/\sqrt{T_K} = M$							
24	ST	23	1005	2^{-1}	M							Store Mach No.

MARK I

Figure 4-7. Compute Mach Number

The numerator .0152 V_p x 2^{-5} must be shifted right one place before it is divided by $\sqrt{T_K}$ since the largest value of the numerator is .5572 and the smallest value of the denominator is .45 x 2^{-5}. Since the numerator is scaled at 2^{-6} after the shift and the denominator 2^{-5}, the resulting M will be scaled at 2^{-1}.

4-32. TIMERS.

4-33. General. A desired elapsed time can be determined in the Mark I by successively subtracting the total time elapsed, each time the storage location passes under the drum read heads, from the total time required for the unit to operate.

4-34. The drum band (fast, medium, or slow) on which the program is to be stored plays a major factor in a timing program. Storage locations on any band pass under the drum read heads at a constant interval (fast band locations every 50 milliseconds, medium band locations every 200 milliseconds, and slow band locations every 800 milliseconds). A numerical constant dependent upon the total time required for the unit to operate as well as the drum band on which the program is to be stored is required in "time delay" programs.

4-35. Figure 4-8 (Part A) is a sample time delay circuit in which the relay will energize 100 seconds after the bus bar has energized and the circuit breaker is "made". In Boolean language, the relay will energize when ABC equals 1. A sample program (when ABC = 1 compute Z = $\sqrt{X^2 + h^2}$) for this time delay circuit, programmed on one of the medium bands is shown in figure 4-8 (Part C). Figure 4-8 (Part B) is a flow chart for figure 4-8 (Part A).

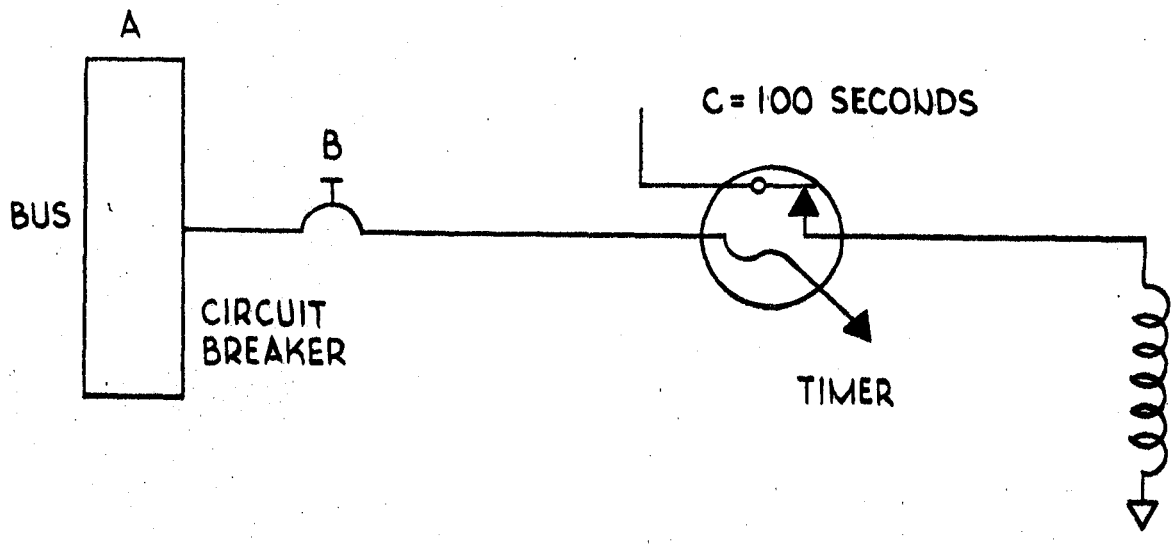


Figure 4-8. Time Delay (Part A - Circuit)

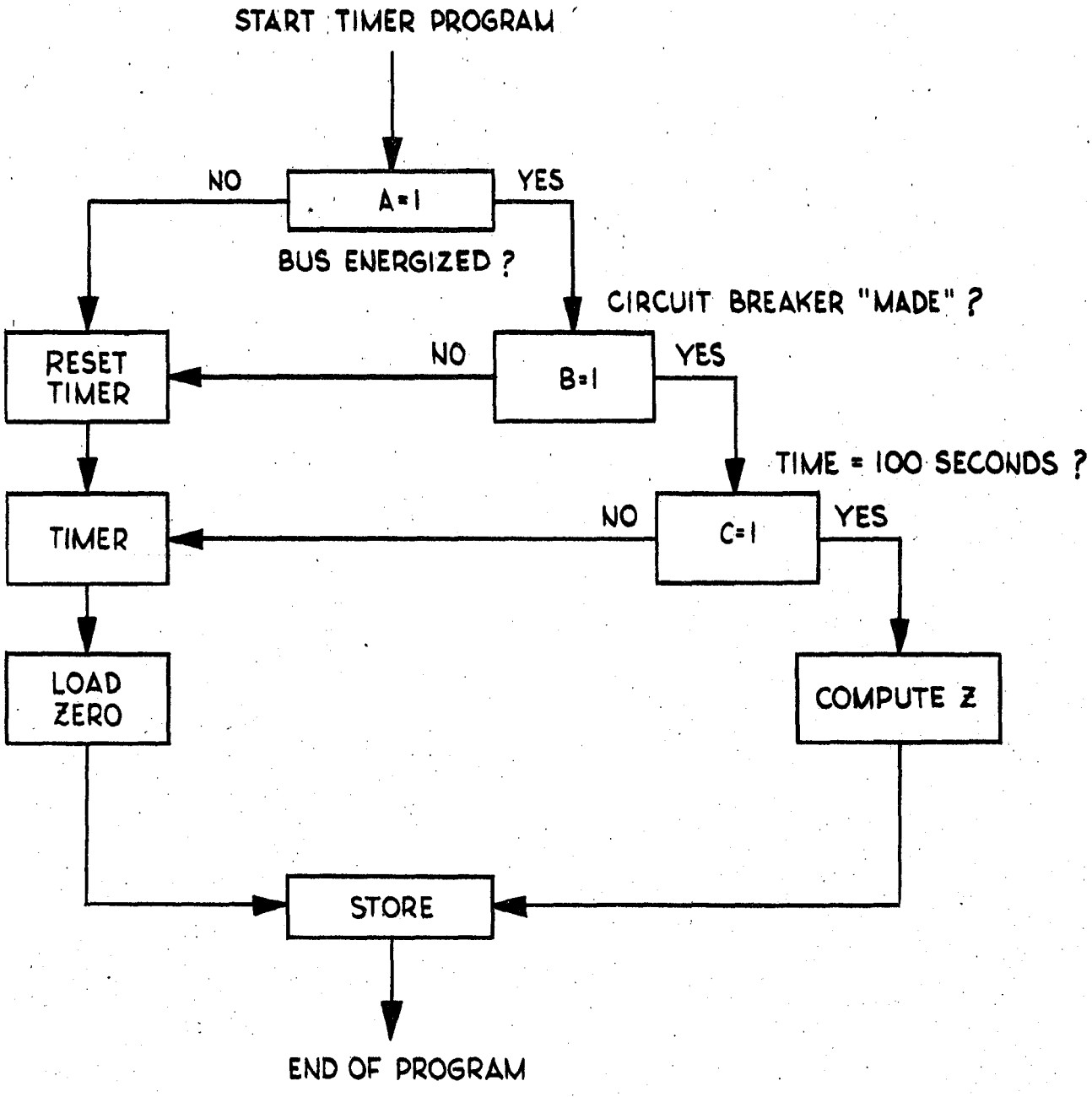


Figure 4-8. Time Delay (Part B - Flow Chart)

DATE 5/9/63

PROBLEM COMPUTE $Z = \sqrt{X^2 + h^2}$ AFTER A TIME DELAY

PROGRAMMER P DAVIS

CODING AND CONSTANT SHEET
MARK I COMPUTER

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
0 1	BLD	3 3	0 1 2 7				A					
0 2	AND	3 1	0 1 2 6				AB					
0 3	SKP	2 6	0 0 0 2									If AB = 0 Steps 04 and 05 are used to reset the timer to 0
0 4	LDK	2 2	0 0 0 0									If AB = 1 Steps 04 and 05 are ignored.
0 5	ST	2 3	1 0 0 4									
0 6	BIN	3 2	0 0 0 0				$\overline{A} + B$					If steps 04 and 05 are read and performed by the machine in-
0 7	SKP	2 6	0 0 3 3									vert and skip to store 0 for Z.
0 8	LDK	2 2	0 0 1 0		500							Part of total time that elapses every time storage location
0 9	ADD	0 1	1 0 0 4		Cn + 500							passes under the drum read heads. Add to previously
1 0	ST	2 3	1 0 0 4		Cn + 1							accumulated time and store for next iteration to advance
1 1	LDK	2 2	7 7 7 6		1							timer.
1 2	SUB	0 2	0 0 0 0		1 - Cn + 1							If answer positive skip and store 0 for Z. If answer negative
1 3	FLN	1 6	0 0 0 0				C					computer Z.
1 4	BIN	3 0	0 0 0 0				$\overline{C}$					
1 5	SKP	2 6	0 0 2 3									
1 6	LD	2 0	1 0 0 0		X							
1 7	SQ	0 5	0 0 0 0									
1 8	NPA	1 2	0 0 0 0									
1 9	NPB	2 5	3 7 7 7									
2 0	NPA	1 2	0 0 0 0									
2 1	NPB	2 5	3 7 7 7		X^2							
2 2	LD	2 0	1 0 0 1		h		lost					
2 3	SQ	0 5	0 0 0 0									
2 4	NPA	1 2	0 0 0 0									
2 5	NPB	2 5	3 7 7 7									
2 6	NPA	1 2	0 0 0 0									
2 7	NPB	2 5	3 7 7 7		h^2							
2 8	ADD	0 1	0 0 0 0		$X^2 + h^2$							
2 9	SRS	0 7	1 0 0 3									
3 0	NPA	1 2	0 0 0 0									

Figure 4-8. Time Delay (Part C - Program) (Sheet 1 of 2)

DATE 5/9/63

PROBLEM COMPUTE $Z = \sqrt{X^2 + h^2}$ AFTER A TIME DELAY

PROGRAMMER P DAVIS

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-82214-A

PAGE 2 OF 2

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
31	NPB	25	3777									
32	NPA	12	0000									
33	NPB	25	3777		$\sqrt{X^2 + h^2}$							
34	BIN	32	0000				C					
35	SKP	28	0001									
36	LDK	22	0000									
37	ST	23	1003									

Figure 4-8. Time Delay (Part C - Program) (Sheet 2 of 2)

4-36. "ILL BEHAVED" FUNCTIONS FOR THE LINEAR INTERPOLATOR.

4-37. General. Many times programmers are faced with the problem of programming function data for the digital interpolator which is not suited to being straight-lined between fixed breakpoints. (The curve has points of inflection whose ordinates do not coincide with any of the fixed breakpoints.) There are various methods of handling this problem, and the most suitable method chosen is determined by the nature of the function curve.

4-38. Warping. An efficient method for use on the Mark I is that of warping the curve into a more reasonable shape by replotting the function data vs. a new variable which is a function of the original independent variable. Instead of plotting f(X) versus X, plot f(X) versus $K_1 X^n + K_2$ or any other deliberately chosen function of X which will warp the function data in the desired manner. Figure 4-9 illustrates four curves. Curve 1 is the original curve of f(X) vs. X. Curve 2 represents a

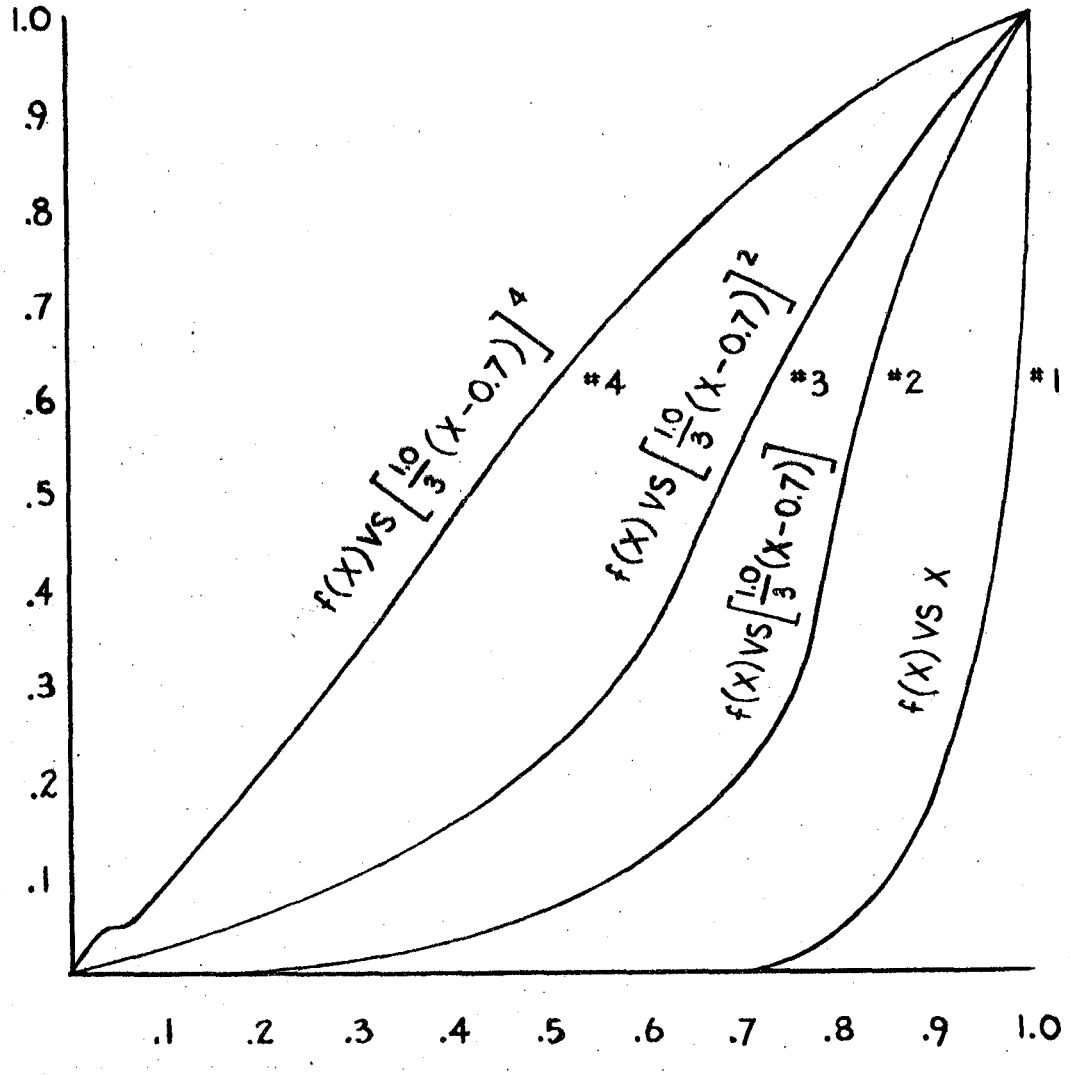


Figure 4-9. Curve Warping

linear "stretching" of the independent variable f(X) vs. $\frac{1.0}{3}(X-.7)$. In curve 3 the functional data is plotted versus $\left[\frac{1.0}{3}(X-.7)\right]^2$ and curve 4 $\left[\frac{1.0}{3}(X-.7)\right]^4$. Any degree of "warping" may be obtained by choosing the proper function of X as an independent variable. Whatever function is chosen as the independent variable must be generated in the general program and stored for use by the digital interpolator.

4-39. The "warping" method is not necessarily a "cure-all" for all of the "ill-behaved" functions which will be encountered by the programmer. It is possible that a function of the independent variable which produces the desired degree of warping is non-existent; or if it does exist, it is too difficult and time-consuming to devise. Further, it is possible that after the proper function has been derived, an excessive number of program steps are required to generate the new independent variable. Splitting a function into more than one curve is an answer to this problem in many cases.

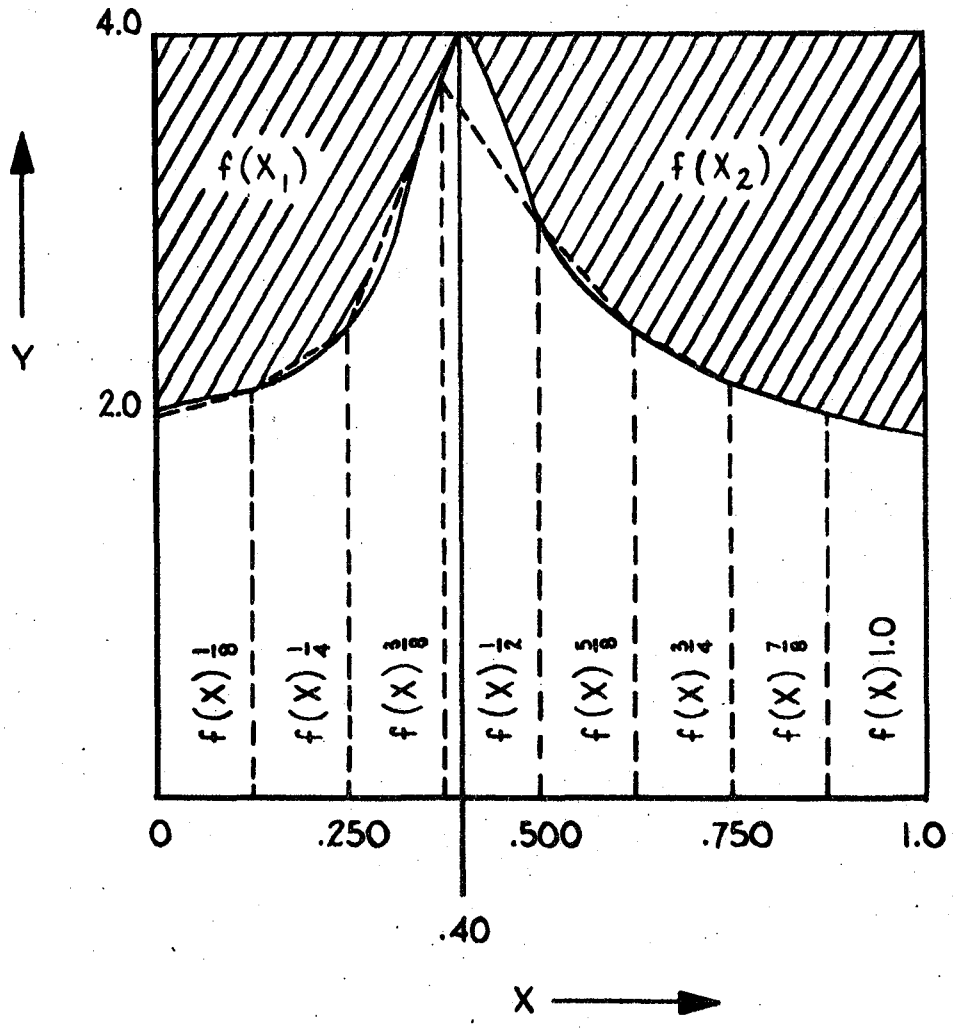


Figure 4-10. "Ill Behaved" Function

4-40. Splitting. Since the nine "breakpoints" are fixed by the logic of the interpolator at intervals of 1/8 of the independent variable, X (i.e., 0, 1,8X, 1/4X, ...X), then the function as constructed by the interpolator will be the dotted line overlaying figure 4-10. f(X) is badly distorted by the removal of the peak in the vicinity of X = 0.40. In order to construct f(X) with accuracy, split f(X) into two functions; $f(X_1)$ for $0 \leq X \leq 0.40$ and $f(X_2)$ for $0.40 < X \geq 1.0$. (See figures 4-11 and 4-12.)

4-41. By rescaling X in the region between X = 0 and X = 0.40, the new quantity X is defined as $X_1 = \frac{X}{0.4}$ for $0 \leqq X \leqq .40$.

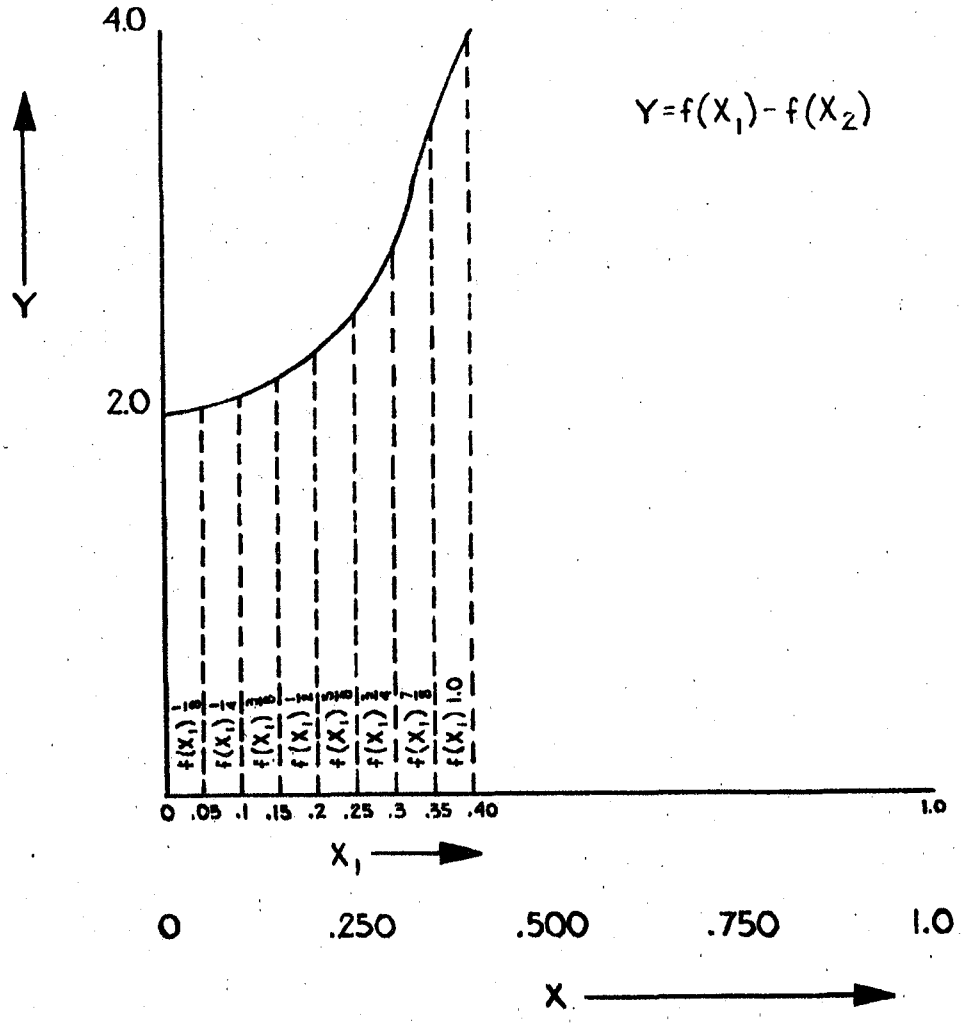


Figure 4-11. Function fX_1

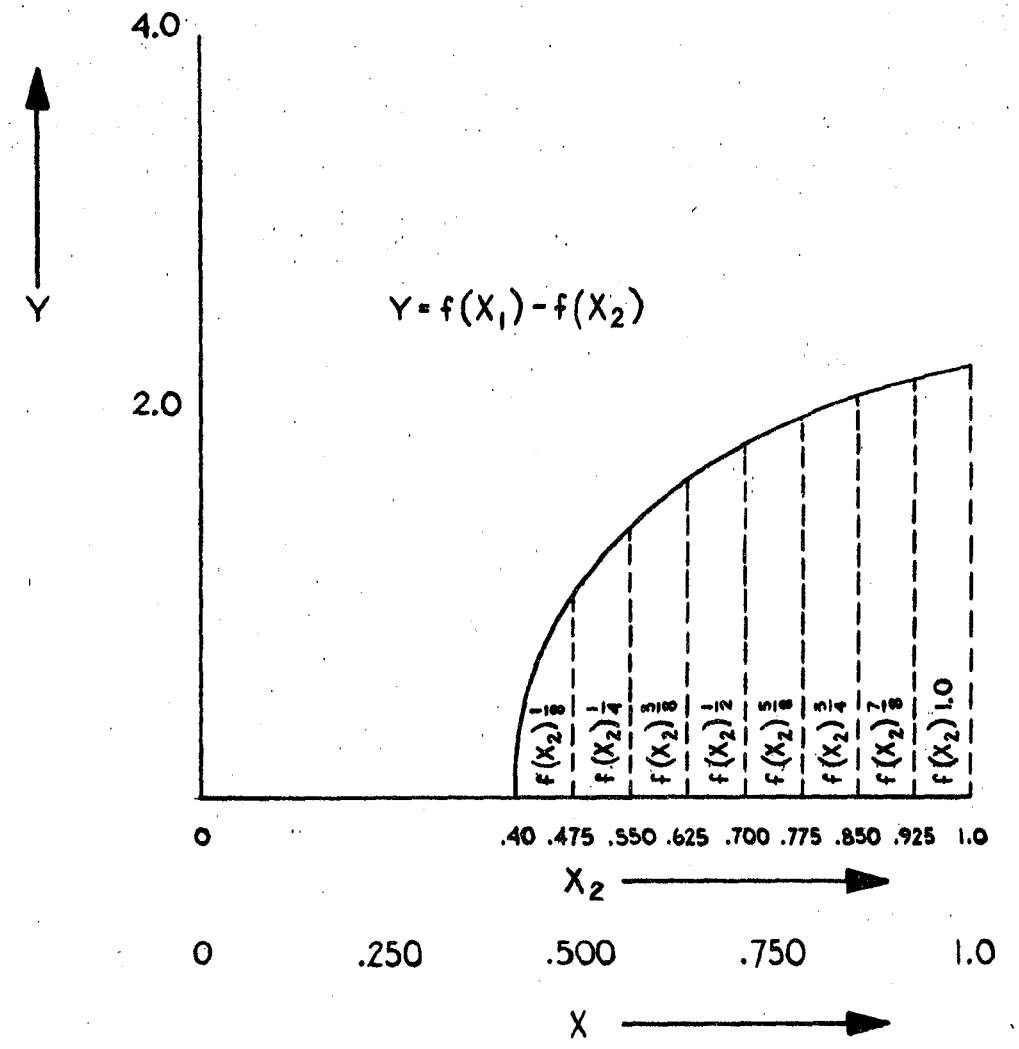


Figure 4-12. Function fX_2

4-42. Rescaling X in the region between X = .40 and X = 1.0 constructs the new quantity X_2 which is defined as: $fX_2 = X - 0.4/0.6$ for $.40 < X \leq 1.0$.

4-43. In figure 4-10 Y = f(X). The new value of Y after "splitting" the function is: $Y = Y_1 = f(X_1) - f(X_2)$.

4-44. The three independent variable (X, X_1, and X_2), must be assigned storage locations in core memory. Core memory locations must also be assigned for the independent variables (Y and Y_1).

4-45. An alternate method of programming X_1 and X_2 is setting a Boolean Flag (Flag negative instruction). Using this method, each time the Mark I calculated X, it would be necessary to test the numerical value of X to see if it was in the region of X_1 or X_2.

4-46. Using a Boolean Flag to program figure 4-10, the data curve for X_1 would remain the same as in figure 4-11 while the data curve for X_2 would be the inverse of figure 4-12 as illustrated in figure 4-13. The Boolean equation for these functions would be:

$$Y = A\, f(X_1) + A^{*}\, f(X_2)$$

Where + signifies or, and * boolean invert.

It is not necessary to use a Boolean Flag type program for any curve used with the 727 simulators.

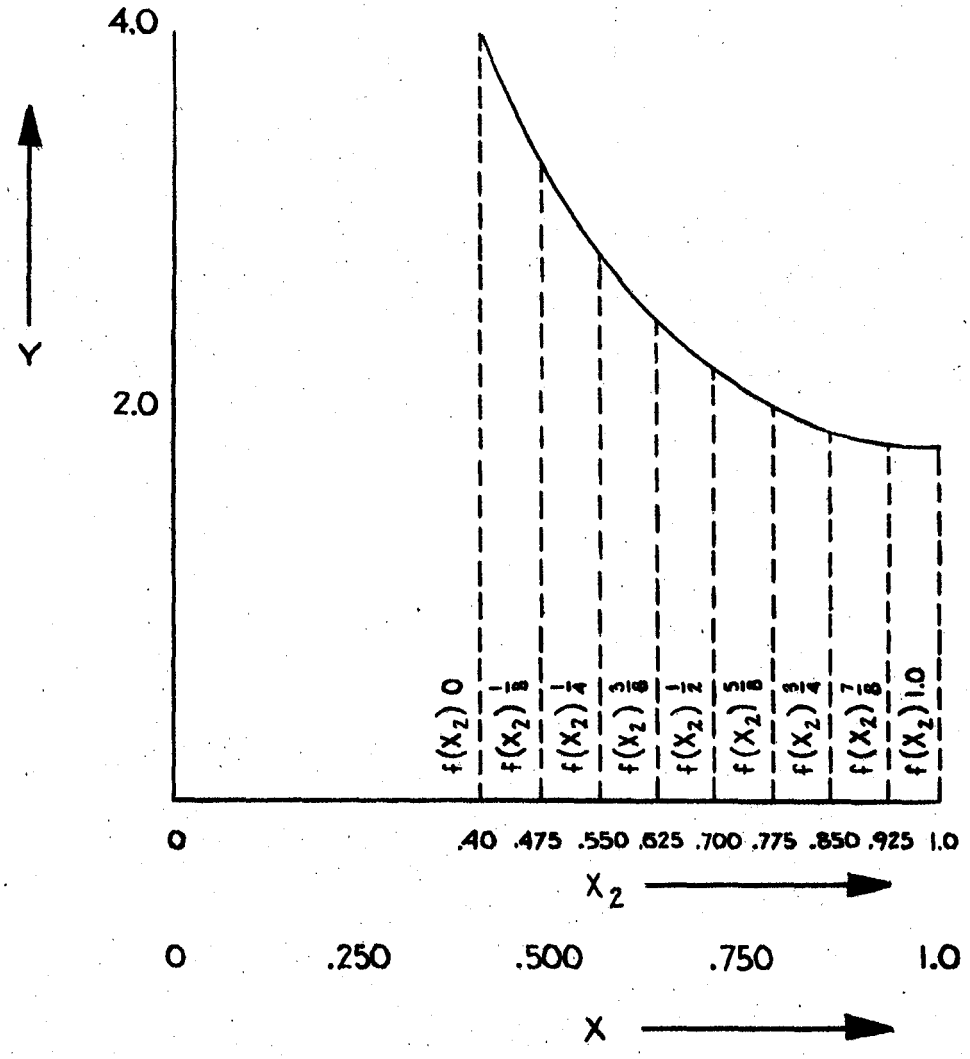


Figure 4-13. Inversion of Figure 4-12

4-47. Coordinate Transfer. In many instances the given data curve for an interpolator function does not lie completely in the first quadrant (i.e., part of the data is numerically negative). Since all negative numbers are treated as zero by the interpolator, the coordinate axis must be "transferred" so that the complete curve lies within the first quadrant.

4-48. Figure 4-14 illustrates a given data curve to be placed on the interpolator section of the drum. To ensure positive values for each of the nine breakpoints, add two (the largest negative number) numerically to the independent variable X, and the dependent variable f(X) which equals Y. Let the new values of X and Y be X_1 and Y_1 therefore $X_1 = X + 2$ and $Y_1 = Y + 2 = f(X_1)$ or $Y = f(X_1) - 2$.

4-49. Plotting X_1 and Y_1, the curve is in the first quadrant (see figure 4-15), and now may be programmed onto the interpolator section of the magnetic drum.

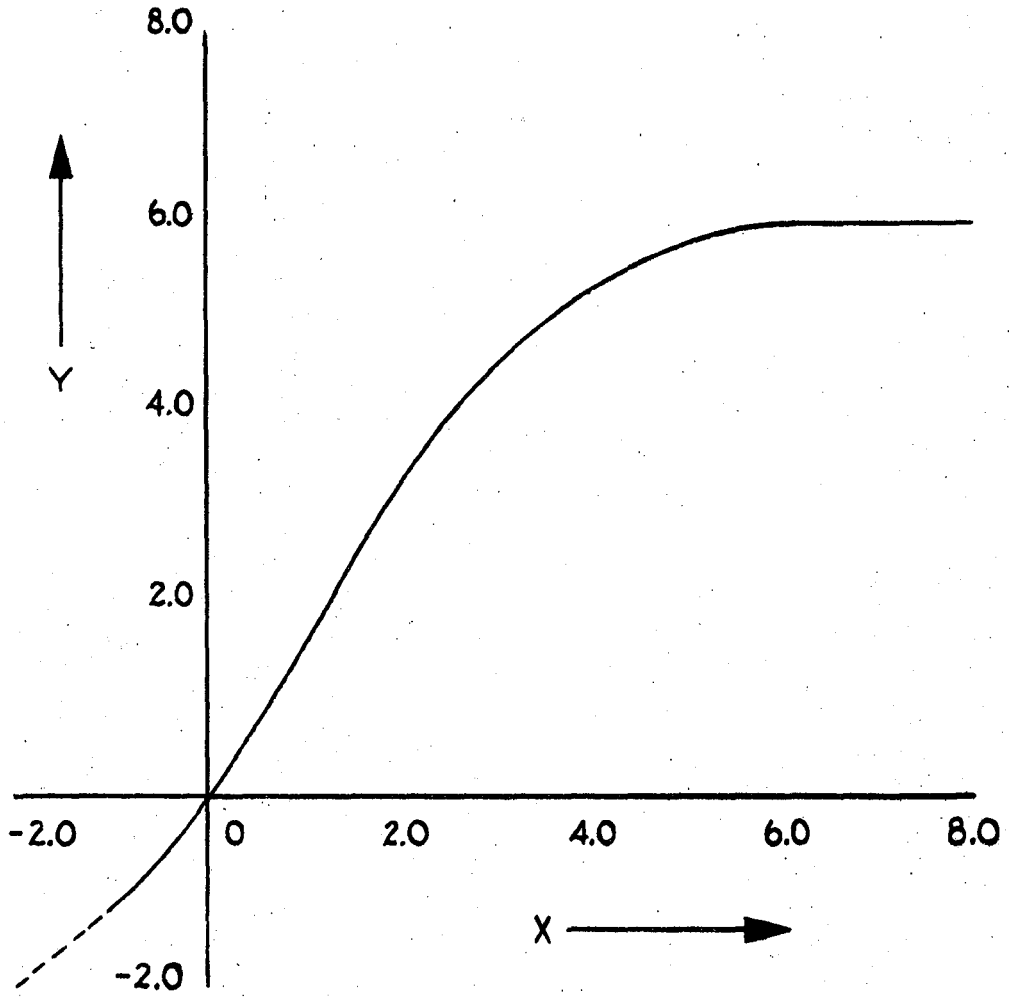


Figure 4-14. Given Data Curve

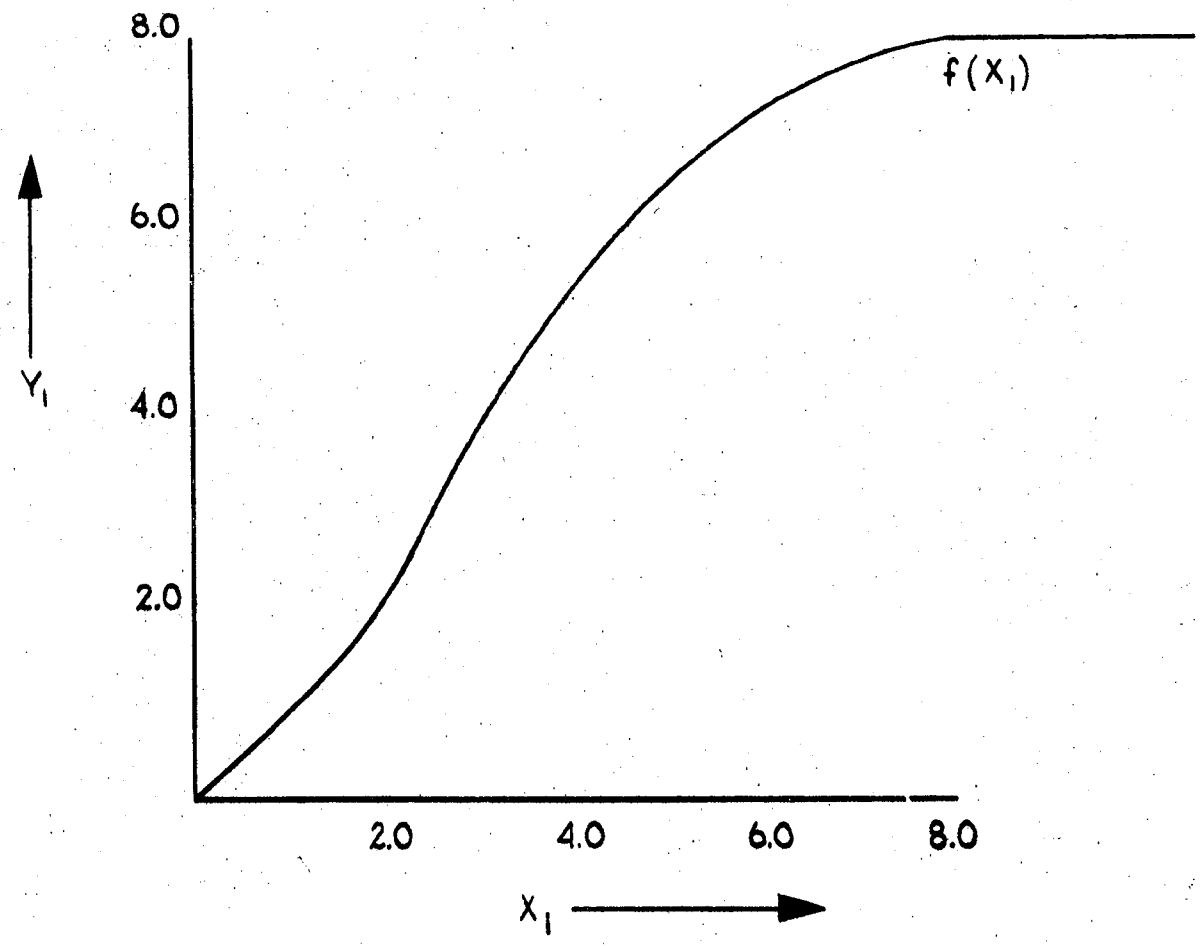


Figure 4-15. Coordinate Transfer of Figure 4-14

4-50. CALCULATION OF ENGINE TRANSIENTS.

4-51. General. Engine transients may occur during start-up, turn-off, change in throttle etc. The example used to explain a transient program will be fuel flow.

4-52. The transient produced in fuel flow (W_f) is caused by a change in throttle position (δth). When the Mark I receives notification that a transient has started because $(\delta th)_{i+1} - (\delta th)_i$ is greater than some constant ϵ, a Boolean flag is set to zero (in the example, flag "A") and a real time clock, t, is set to zero. In later iterations, the clock, t, will be incremented by some constant amount depending on the band this program is written on (refer to paragraph 4-34), until the program determines that the elapsed time from the start of the transient to the present time is greater than a specified amount. The flag will then be set to the non-transient case and the transient will be considered finished. However, the program will continue checking for changes in throttle position $(\delta th)_{i+1} - (\delta th)_i$ to determine if new transients are starting. The program will also consider a difference of sign between the quantities $[(\delta th)_{i+1} - (\delta th)_i]$ or $[(\delta th)_i - (\delta th)_{i+1}]$ as the start of a transient.

4-53. The program will compute a W_f that will reflect the transient conditions by means of a multiplying constant $\epsilon - \lambda t^2$. This function will be written on the interpolator section of the drum. The value of W_f when the clock was set to zero will be used as the value of W_f for $t = 0$. W_{fc} will be computed using the regular equation for this quantity. The equation used for the transient case will be: $W_f = W_{fc} + (W_{f_{t=0}} - W_{fc})$. For the non-transient case, $W_f = W_{fc}$.

4-54. The flow chart (figure 4-16 Part (A)), and the "Coding and Constant Sheet" (figure 4-16 Part (B)) describe the techniques used. This problem is only intended as an example of programming engine transients, but drawing a flow chart before coding commences is almost absolutely necessary in problems of this type.

4-55. DOUBLE NUMBERS (1 WORD = 2 NUMBERS).

4-56. General. Double numbers will be encountered in problems pertaining to the radio aids portion of the program. Since one word in the radio aids section consists of 20 bits, and these 20 bits may sometimes describe two pieces of data; e.g., the most significant ten bits describe the X coordinate of a radio aid, while the least significant ten bits describe the Y coordinate. When this 20 bit word is loaded into the accumulator, the last four bits of the accumulator are zeros.

4-57. If operations are to be performed with the X coordinate, a right shift of one place clears the number out of the sign location and makes the number positive. The presence of the ten Y bits can only cause an error in the LSB of X. When operations are to be performed on Y, it is necessary to shift the accumulator left nine places, and make the result positive. This may be accomplished by two successive left shift commands, followed by an absolute value command to make the sign of the number positive.

4-58. To form a 20 digit number from two separate ten digit numbers in two different Core Memory locations, load the accumulator with the number that is to form the least significant part of the result, and make the sign positive with an absolute value command. Scale the number right ten places by two successive scale right commands, then scale right three places and left three places to eliminate the three least significant bits. The contents of the accumulator should be as shown in figure 4-17.

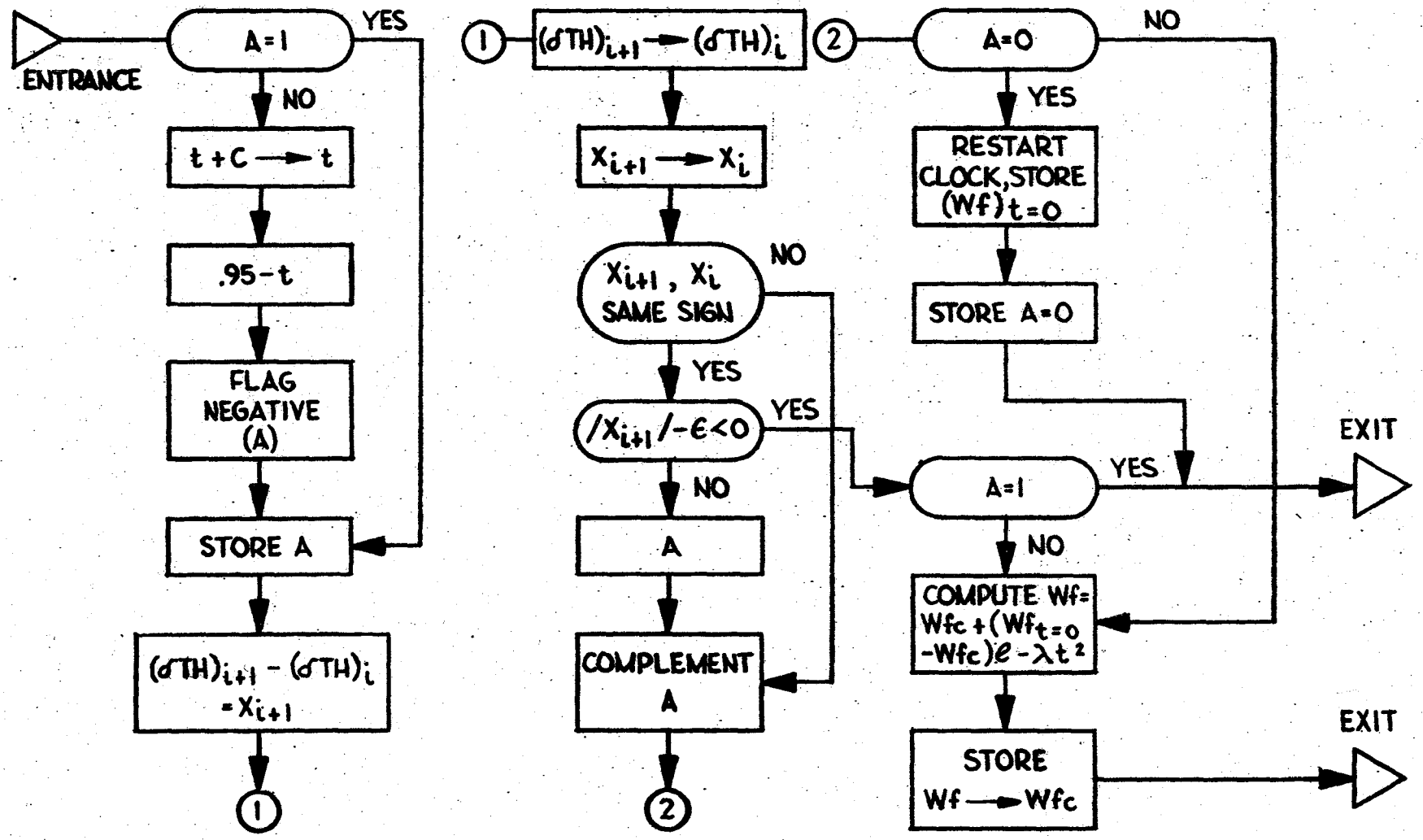


Figure 4-16. Wf - Transient Part A - (Flow Chart)

DATE JUNE 3, 1963

PROBLEM Wf TRANSIENTS

PROGRAMMER J. DOE

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-2214-A

PAGE 1 OF 2

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	BLD	33	3777				A					
02	SKP	26	0004									
03	LDK	22	7463		.95							
04	SUB	02	1000		.95 - t							
05	FLN	16	0000									
06	BST	34	3777									
07	LD	20	1001		$(\delta^{TH})_{i+1}$							
08	SUB	02	0500		$(\delta^{TH})_{i+1} - (\delta^{TH})_{i} = X1_{i+1}$							
09	ST	23	1017									Store X1 + 1
10	LD	20	1001		$(\delta^{TH})_{i+1}$		⑤ lost					
11	ST	23	0500		$(\delta^{TH})_{i+1}$ $(\delta^{TH})_{i}$							
12	LD	20	1016		X1		⑤ lost					
13	FLN	16	0000									
14	BST	34	3776				W					Store Sign Of X1
15	LD	20	1017		$X1_{i+1}$		⑤ lost					
16	ST	23	1016		$X1_{i+1} \rightarrow X1$							
17	FLN	16	0000									
18	BLD	33	3775				Y					Store Sign Of X1 + 1
19	BIN	32	0000				Y					
20	AND	31	3776				Yw					
21	BLD	33	3776				W					
23	BIN	32	0000				W					
23	AND	31	3775				Wy					
24	OR	30	0000				WY + YW					
25	SKP	26	0010									
26	ABS	14	0000		$/X1_{i+1}/$							
27	LDK	22			ϵ		⑤ lost					
28	DIS	13	0000		$-\epsilon$							
29	ST	23	1015									Store For Later Zero Slice
30	ADD	01	0000		$-\epsilon + /X1_{i+1}/$							

Figure 4-16. $\overline{Wf}$ - Transient Part B - Program (Sheet 1 of 2)

DATE JUNE 3, 1963

PROBLEM WI TRANSIENTS

PROGRAMMER J. DOE

CODING AND CONSTANT SHEET

MARK I COMPUTER

PAGE 2 OF 2

F-2214-A

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
31	FLN	16	0000									
32	SKP	26	0013									
33	BLD	33	3777				A					
34	BIN	32	0000				$\overline{A}$					
35	SKP	26	0012									
36	BST	34	3777				A = 0					
37	LD	20	1001		Wft = 0		lost					
38	ST	23	1002									Store Wf at t = 0
39	LD	20	1015		$-\epsilon$		lost					
40	FLN	16	0000									
41	ZSL	15	0000									
42	ST	23	1000		$0 \rightarrow t$							Zero Clock
43	SKP	26			$\longrightarrow$							Exit
44	BLD	33	3777				A					
45	SKP	26			$\longrightarrow$							Exit
46	LD	20	1002		Wf at t = 0		lost					
47	SUB	02	1001		(Wft = 0 - Wfc)							
48	MLT	03	1170		(Wft = 0 - Wfc) $e^{-\lambda t^2}$							
49	NPA	12	0000									
50	NPB	25	3777									
51	NPA	12	0000									
52	NPB	25	3777									
53	ADD	01	1001		Wfct (Wft = 0 - Wfc) $e^{-\lambda t^2}$							
54	ST	23										Store Wf

MARK I

Figure 4-16. Wf - Transient Part B - Program (Sheet 2 of 2)

0 | 0000000000 | YYYYYYYYYY | 000

Figure 4-17. Ten Least Significant Bits

4-59. Load the accumulator with the number that is to form the most significant part, scale right 13 places and then left 13 places, make the sign positive with an absolute value command, and add the contents of the salvage register. The contents of the accumulator will now be as shown in figure 4-18.

0 | XXXXXXXXXX | YYYYYYYYYY | 000

Figure 4-18. 20 Bit Number

4-60. The absolute value instructions may be eliminated if it is known that the values of X and Y are positive.

4-61. HANDLING A NUMBER AS A "COARSE" AND "FINE" SUM (1 NUMBER = 2 WORDS).

4-62. General. The situation in flight simulators in which some variable with an extremely large magnitude (e.g., distance, altitude) is calculated by integrating its derivative will be very often encountered by the programmer. With 23-bit resolution in the Mark I, the resolution of the variable may be adequate despite its great magnitude. It must be noted however, that the smallest bit (i.e., the 24th bit) of the word representing that variable may in itself represent a number which is fairly large in magnitude. The magnitude of this least significant bit, defines the smallest increment that may be added to the variable concerned.

4-63. Integration. Integration in the Mark I is accomplished by calculating an incremental change in the quantity concerned and adding that increment to the old value of the quantity. In the equation $X = \int \dot{x}\, dt$, if X is extremely large in magnitude, then the LSB of X is correspondingly large, and the magnitude of this LSB is the smallest rate of change of X which can be integrated with accuracy. (A smaller $\dot{X}$ would need a smaller increment to add to X.)

4-64. If it is necessary to "find a smaller increment" to add to X, so that very small rates can be integrated accurately, then it is necessary to magnify the LSB of X, thus improving the resolution.

4-65. This situation is handled by treating X as the sum of two numbers: X = X course (Xc) + X fine (Xf). Xc will have the same scale factor as X ordinarily would. $\dot{X}$ will be integrated to generate Xf, where Xf is 2^{10} Xc. This represents a ten-bit increase in resolution. Each time that Xf is calculated by integrating $\dot{X}$, it is tested to see if it is large enough to add a significant digit to Xc. If it is large enough, Xf is scaled right ten places and added to Xc. Whatever amount is added to Xc must be subtracted from Xf. A sample problem of this nature is illustrated in figure 4-19 where:

DATE 5/29/63

PROBLEM "COARSE" AND "FINE" SUM

PROGRAMMER J. JAMES

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-2214-4

PAGE 1 OF 1

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	BALV.	BOOLEAN ACCUMULATOR	BALV. 1	BALV. 2	BALV. 3	BALV. 4	REMARKS
				2^{-16}	ΔX_F							
0 1	SUB	0 2	0 1 0 2	2^{-16}	$\Delta X_F - (X_F)$							$\Delta X_F + X_F$ = New X_F
0 2	ST	2 3	0 1 0 3	2^{-16}	X_F							Temporarily Store X_F
0 3	SCL	1 0	1 0 0 5	2^{-21}	X_F							
0 4	SCL	1 0	1 0 0 5	2^{-26}	X_F							
0 5	ST	2 3	0 1 0 4	2^{-26}	X_F							Temporarily Store X_F
0 6	ADD	0 1	0 1 0 1	2^{-26}	$X_F + X_C = X_C$							Calculate X_C and Store
0 7	ST	2 3	0 1 0 1	2^{-26}	X_C							
0 8	LD	2 0	0 1 0 4	2^{-26}	X_F							
0 9	SCL	1 0	0 0 0 5	2^{-21}	X_F							Adjust X_F and Store
1 0	SCL	1 0	0 0 0 5	2^{-16}	X_F							
1 1	SUB	0 2	0 1 0 3	2^{-16}	$-X_F$							
1 2	ST	2 3	0 1 0 2	2^{-16}	$-X_F$							

MARK I

Figure 4-19. "Coarse" and "Fine" Sum Program

Xc is scaled at 2^{-26} and stored in location 0101

-Xf is scaled at 2^{-16} and stored in location 0102

locations 0103 and 0104 are used for temporary storage

ΔXf is scaled at 2^{-16}

Assume ΔXf has been calculated and is now stored in the accumulator.

4-66. LATITUDE-LONGITUDE INFORMATION.

4-67. All radio calculations will use aircraft location stored in units of degrees of Latitude (λ) and Longitude (L) scaled for the continental United States. (See table 4-1.) Figure 4-20 shows the sign convention to be used for the United States.

Table 4-1. Latitude-Longitude Scale

Bit	Degrees	Nautical Miles	Feet
1	±64	3,840	23,348,239.3
2	32	1,920	11,679,119.65
3	16	960	5,839,559.825
4	8	480	2,919,779.913
5	4	240	1,459,889.956
6	2	120	729,944.978
7	1	60	364,972.489
8	30'	30	182,486.245
9	15'	15	91,243.122
10	7'30"	7.5	45,621.561
11	3'45"	3.75	22,810.781
12	1'52.5"	1.875	11,405.390
13	56.25"	0.9375	5,702.695

Bit	Degrees	Nautical Miles	Feet
14	28.125"	0.46875	2,851.3475
15	14.0625"	0.234375	1,425.674
16	7/03125"	0.1171875	712.837
17			356.418
18			178.209
19			89.104
20			44.552
21			22.276
22			11.138
23			5.569
24			2.784

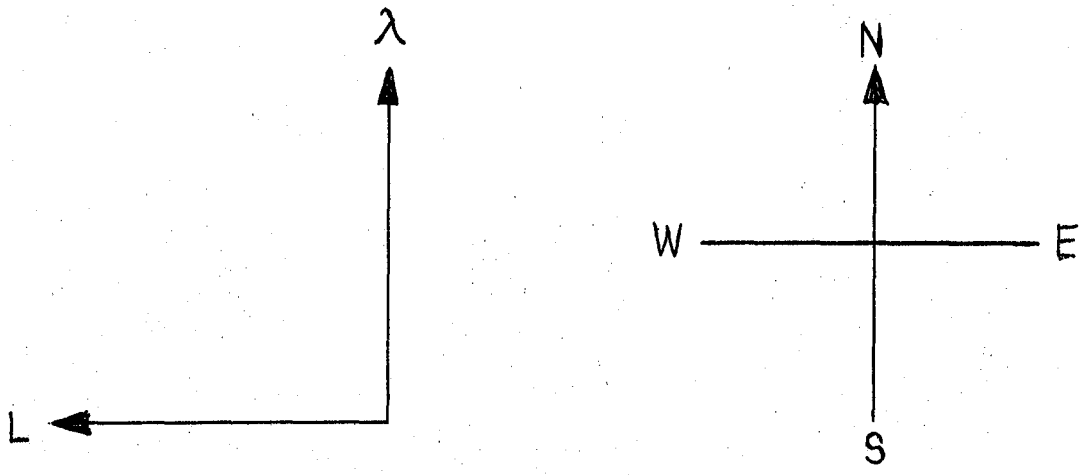


Figure 4-20. Latitude-Longitude Sign Convention

4-68. The values of λ and L are derived from the following equations.

$$\text{Latitude} = \lambda = (2.7411\text{X}10^{-6}) \int \left[UL_1 + VM_1 + WN_1 + vW_1\right]$$

$$\text{Longitude} = L = (2.7411\text{X}10^{-6}) (\sec\lambda) \int \left[UL_2 + VM_2 + WN_2 + vW_2\right]$$

Where: L_1 L_2, M_1 M_2, are direction cosines

U, V, and W are translational velocities along the X, Y, and Z areas respectively.

vW_1 and vW_2 are wind velocity components.

The constant $2.7411\text{X}10^{-6}$ is the conversion factor for converting feet to degrees of arc.

4-69. The equations used for converting degrees to nautical miles follow:

$$\lambda' = 60\lambda$$

$$L' = 60L \cos\lambda$$

where: λ = latitude in degrees of arc

λ' = distance along any meridian in nautical miles (NM)

L = longitude in degrees of arc

L' = distance along the parallel in nautical miles

4-70. COORDINATE SYSTEMS.

4-71. General. Many different coordinate systems are needed for navigation calculations. The basic system will be LATITUDE-LONGITUDE (paragraph 4-66). Other coordinate systems used are the X - Y and a-c rectangular systems and the R - ψ polar system.

4-72. X-Y Rectangular System. The X - Y rectangular system will be used for recorder applications and in small areas where meridian convergence can be neglected. The system is a rectangular system and the X axis always points to TRUE NORTH (see figure 4-21).

4-73. a-b Rectangular System. The a-b rectangular system is used in marker beacon and A-N range calculations for the rotation of the X - Y coordinate system through an angle ψ where $-90^{\circ} > \psi > +90^{\circ}$. Figure 4-22 illustrates the sign convention to be used.

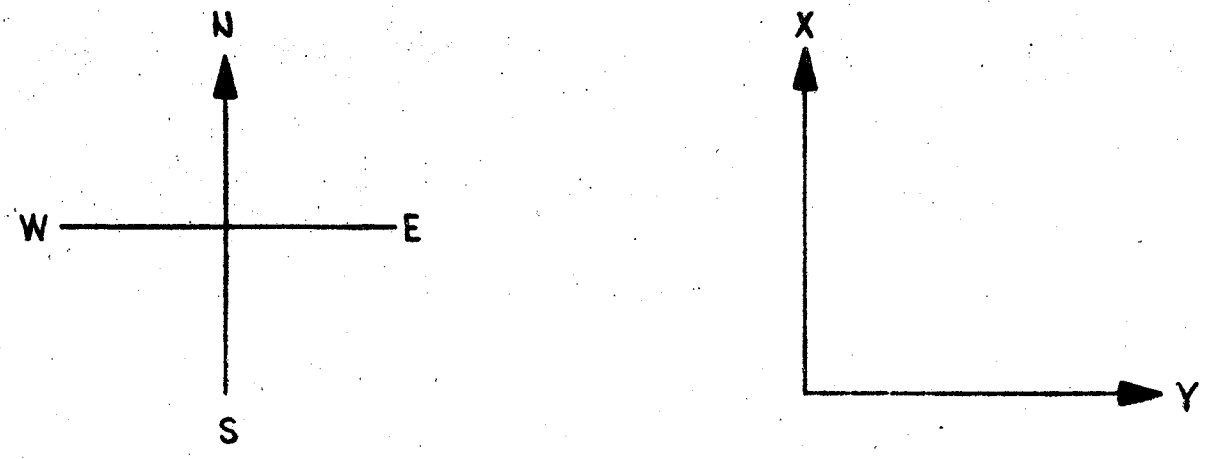


Figure 4-21. X - Y Coordinate System Sign Convention

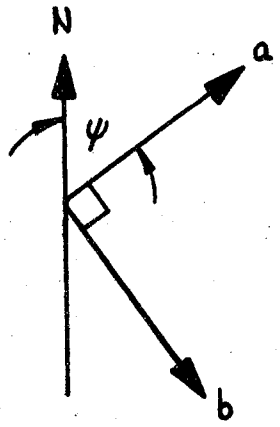


Figure 4-22. a-b Coordinate System Sign Convention

4-74. R - ψ Polar System. The R - ψ polar system is used for bearing and range systems such as in VOR and DME. The sign convention is illustrated in figure 4-23.

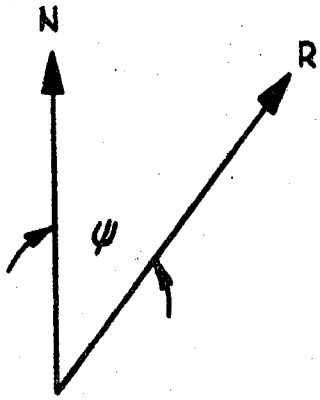


Figure 4-23. R - ψ Polar System Sign Convention

4-75. Converting From L- λ to a-b Coordinates.

4-76. For small distances we can assume the L- λ coordinate system to be truly rectangular, thus when converting from a L- λ system to the a-b system (refer to figure 4-24), the following equations can be used:

$$a_{GAS} = 364{,}816 \left[(Ls - La)\cos\lambda a \sin\psi + (\lambda a - \lambda s)\cos\psi\right]$$

$$b_{GAS} = 364{,}816 \left[(Ls - La)\cos\lambda a \cos\psi - (\lambda a - \lambda s)\sin\psi\right]$$

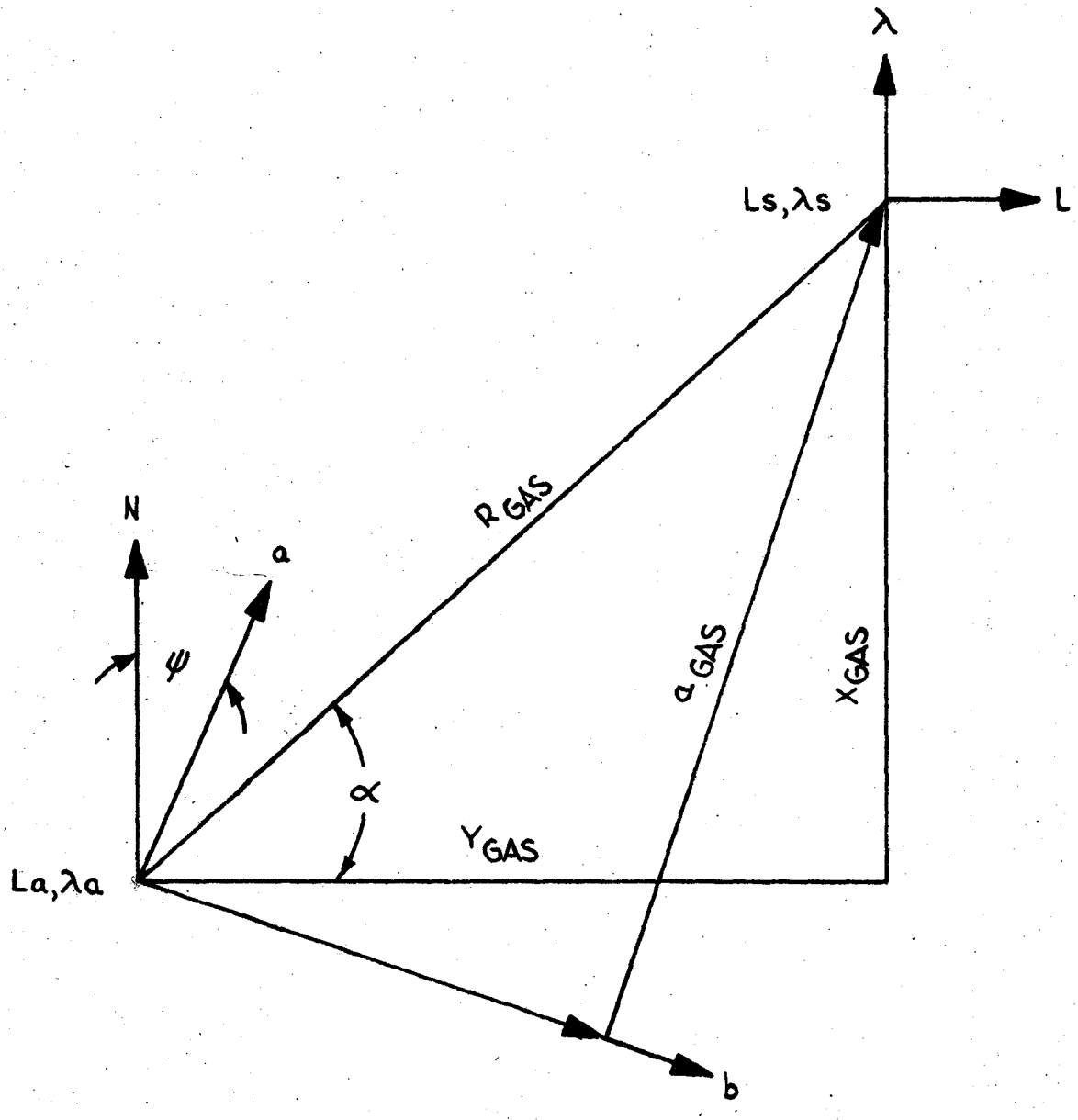


Figure 4-24. Converting From L- λ To a-b Coordinates

where L and λ are degrees

cos λ a is a function from the LFI

sin ψ and cosine ψ are station data words

a_{GAS} and b_{GAS} are in feet

Subscripts - a is for aircraft
s is for station

Derivation

$$a_{GAS} = R_{GAS} \sin(\psi + \alpha)$$

$$= R_{GAS} [\sin\psi \cos\alpha + \cos\psi \sin\alpha) \quad (1)$$

$$b_{GAS} = R_{GAS} \cos(\psi + \alpha)$$

$$= R_{GAS} [\cos\psi \cos\alpha - \sin\psi \sin\alpha) \quad (2)$$

$$\cos\alpha = Y_{GAS}/R_{GAS} \quad (3)$$

$$\sin\alpha = X_{GAS}/R_{GAS} \quad (4)$$

Substituting equations (3) and (4) in equation (1)

$$a_{GAS} = R_{GAS} [\sin\psi\ Y_{GAS}/R_{GAS} + \cos\psi\ X_{GAS}/R_{GAS}]$$

$$a_{GAS} = Y_{GAS} \sin\psi + X_{GAS} \cos\psi \quad (5)$$

In like manner equation (2) becomes

$$b_{GAS} = Y_{GAS} \cos\psi - X_{GAS} \sin\psi \quad (6)$$

$$X_{GAS} = 364{,}816\ (\lambda a - \lambda s) \quad (7)$$

$$Y_{GAS} = 364{,}816\ (Ls - La) \cos\lambda a \quad (8)$$

where: 364,816 is the conversion factor for converting degrees to feet.

Substituting equations (7) and (8) in equations (5) and (6)

$$a_{GAS} = 364{,}816 [(Ls - La) \cos\lambda a \sin\psi + (\lambda a - \lambda s) \cos\psi] \quad (9)$$

$$b_{GAS} = 364{,}816 [(Ls - La) \cos\lambda a \cos\psi - (\lambda a - \lambda s) \sin\psi] \quad (10)$$

4-77. LAMBERT CONFORMAL CONIC PROJECTION CALCULATIONS.

4-78. The recorder plots in an X - Y coordinate system on a Lambert Conformal Conic Projection Chart. Because it plots in an X - Y system it is necessary to convert the aircraft location in the L-λ coordinate system into an X - Y system for the recorder.

4-79. The following equations are to be used for the computation of the X - Y coordinate for the recorder. (Refer to figures 4-25 and 4-26.)

$$\gamma = (L_a - L_0) \sin \phi \qquad (1)$$

$$\rho a = \rho 1 + 60 (\lambda a - \lambda 1) + f(\lambda) \qquad (2)$$

$$\rho 0 = \rho 1 = 60 (\lambda 0 - \lambda 1) + f(\lambda) \qquad (3)$$

$$X_0 = 72913.2/\mu \rho 0 \qquad (4)$$

$$X_c = 72913.2/\mu (-\rho a) \cos \gamma + X_0 \qquad (5)$$

$$Y_c = 72913.2/\mu \rho a \sin$$

where:

L = longitude (degrees)

λ = latitude (degrees)

ρ = radius of cone (NMI)

γ = degrees

X = X coordinate (inches)

Y = Y coordinate (inches)

μ = scale factor (e.g., with a 1:2,000,000 scale μ = 2,000,000)

72913.2 = number of inches per nautical mile

Subscripts:

0 pertains to chart reference points of (0, 0)

1 and 2 pertains to standard parallel of projection where subscript 1 is for the greater value

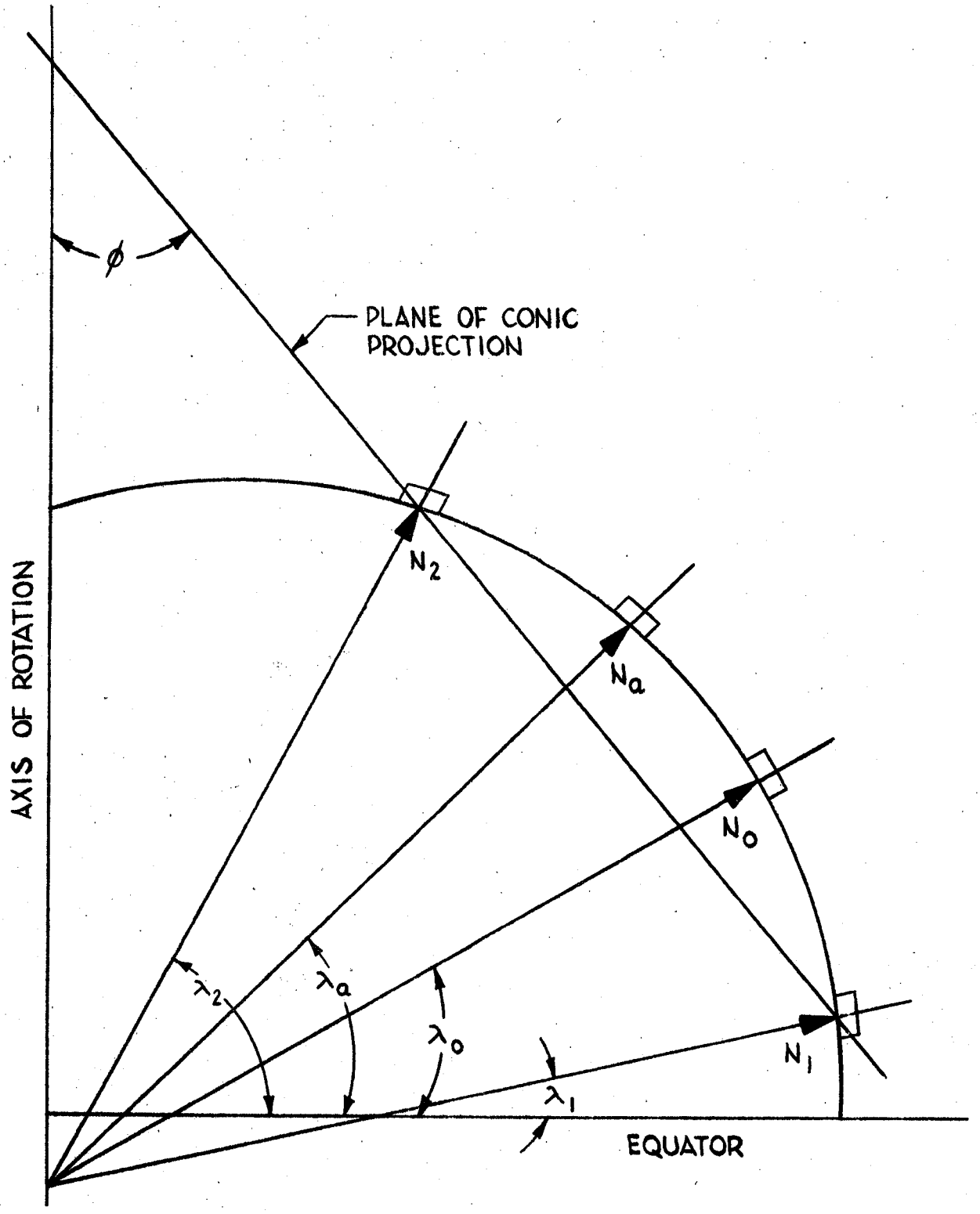


Figure 4-25. Lambert Conic Projection, Plane Perpendicular To Projection

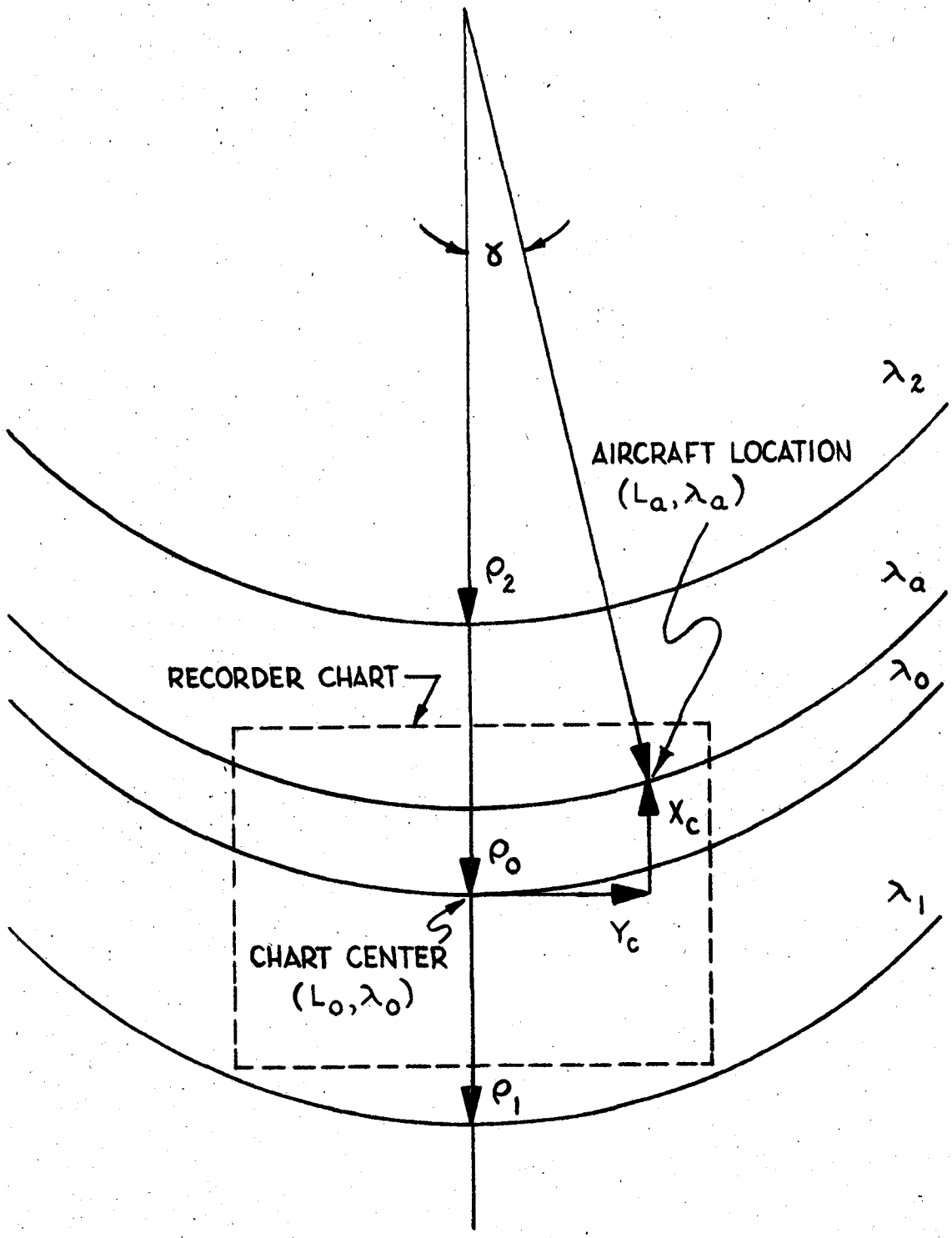


Figure 4-26. Lambert Conic Projection, Chart Plane

a pertains to aircraft location
c pertains to chart location of pen

$\sin \phi$, ρ, and $f(\lambda)$ are emperical functions unique to the standard parallels for the projection used. These terms are explained in paragraph 4-77.

4-80. EMPERICAL FUNCTIONS.

4-81. The cone of projection makes an angle ϕ with the axis of rotation of the earth. The $\sin \phi$ the sine of the angle also considers the eccentricity of the earth. The equation for the sine of ϕ follows:

$$\sin \phi = \frac{\log \cos \lambda 1 - \log \cos \lambda 2 + \log N_1 - \log N_2}{\log \tan \left(\frac{\pi}{4} + \lambda 2\right) - \log \tan \left(\frac{\pi}{4} + \frac{\lambda 1}{2}\right)}$$

$$N_1 = \frac{a}{\sqrt{1 - E^2 \sin^2 \lambda_1}}$$

$$N_2 = \frac{a}{\sqrt{1 - E^2 \sin^2 \lambda_2}}$$

ρ where E = eccentricity = 0.0822751

a = radius of earth = 3437.75 NM

ρ_1 is the radius of the concentric circle in NMI, representing λ_1 on the chart projection.

$$\rho 1 = \frac{N_1 \cos \lambda_1}{\phi \sin}$$

The values of $\sin \phi$ and ρ 1 for the standard Lambert Projections used with the TWA 727 is as follows:

Standard Parallels	ρ 1	ϕ sin
33/45	4591.524 NMI	0.627926

4-82. $f(\lambda a)$ is a correction factor added to equations (2) and (3) (paragraph 4-77), to increase the accuracy of the radii since the 60 (NM/Degrees) is an approximation. The exact value of ρ can be calculated by the following equation. Having the exact value of ρ, the correction factor $[f(\lambda a)]$ can be calculated and plotted as a function of λ a.

$$\ln r = -\sin \phi \ln \left[\tan \frac{\pi}{4} + \frac{\lambda}{2}\right] + C_i$$

where $C_i = \ln \rho_1 + \sin \phi \ln \left[\tan \left(\frac{\pi}{4}\right) + \left(\frac{\lambda}{2}\right)\right]$

r = the exact radius of ρ at λ

λ = latitude of point in question

Therefore

$$f(\lambda) = r = \rho_1 - 60 (\lambda - \lambda_1)$$

4-83. MARKER BEACONS.

4-84. All types of 75 MC marker beacons are simulated by using the following equations:

a. FM $E = 695h/8a^2 + b^2 + 8h^2$

b. BONE $E = 600h/8a^2 + b^2 + 4h^2 \left[(0.5 + \left|\frac{b}{a}\right|\right]$ where $\left|\frac{b}{a}\right| \leq 1$

c. ILS $E = 64h/8a^2 + b^2 + h^2$

d. LFM $E - 20h/2a^2 + b^2 + h^2/4 - \frac{80h}{8a^2 + 4b^2 + h^2}$

Z $E = 72h/8a^2 + 8b^2 + h^2$

In the above equations:

a = distance from station to aircraft along the minor axis in feet.

b = distance from station to aircraft along the major axis in feet.

h = altitude of the aircraft above the station in feet.

E = signal level at the aircraft in volts.

4-85. LFRR CALCULATIONS.

4-86. General. The four "Coarse Legs" making up the LFRR patterns will be simulated through the use of four circular radiation patterns (see figure 4-27).

4-87. Opposite pairs of circles represent the "A" and "N" signal, the centers of these circles form an "a-b" rectangular coordinate system. The "Course Legs" are the areas where the circles overlap. In order to provide the necessary combination of "Course Legs" it is necessary to provide rotation of the rectangular coordinate system (see figure 4-28). This angle of rotation is always measured from "TRUE NORTH" and the a axis.

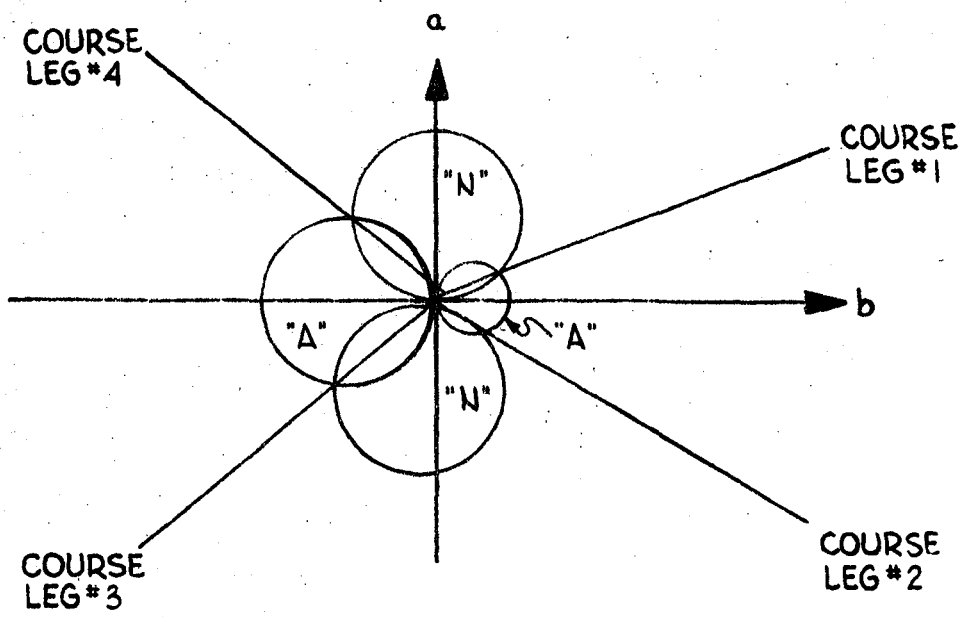


Figure 4-27. A-N Course Legs

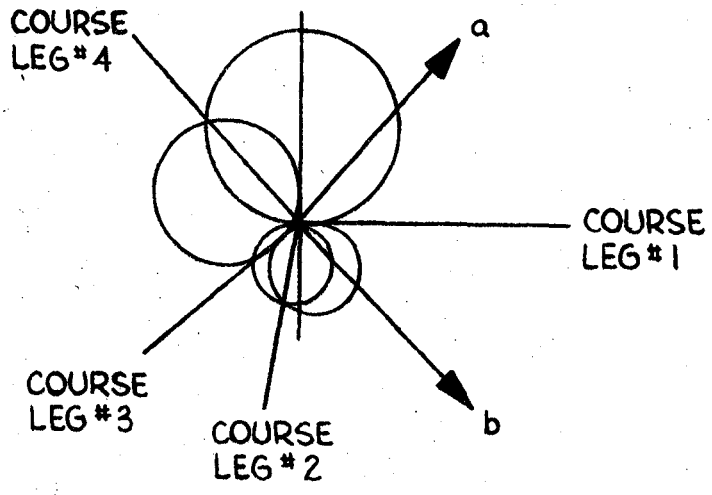


Figure 4-28. A-N Range Rotation

4-88. A-N Audio. To simulate the A-N audio, a servo with "A" audio on one end of a potentiometer and "N" audio on the other end of the potentiometer (figure 4-29) is required.

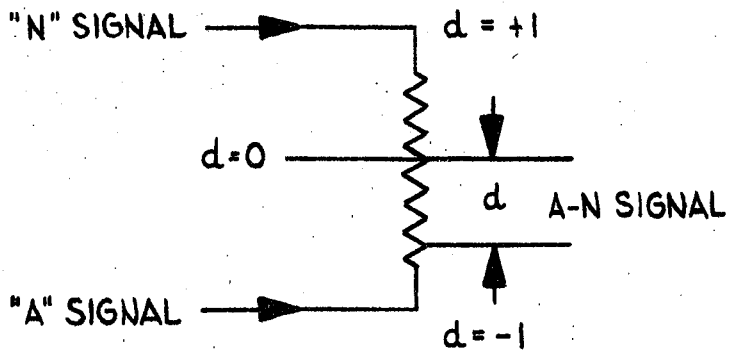


Figure 4-29. A-N Range Audio Servo

4-89. The "A-N Range Audio" servo will have an input which represents the "A" to "N" signal ratio. If d is the servo displacement from center, then $d_{max} = +1$ and $d_{min} = -1$. Solving for d as a function of n and a, the following equations are derived:

$$n = \frac{1 + d}{2}$$

$$a = \frac{1 - d}{2}$$

$$\frac{a}{n} = \frac{1 - d}{1 + d}$$

$$a + da = n - dn$$

$$d = \frac{n - a}{n + a} \qquad (1)$$

where a and n are the signal levels of the A-N signals of the aircraft.

4-90. Assuming that the "a - b" coordinates to the aircraft have been solved (paragraph 4-73), refer to figures 4-30 and 4-31.

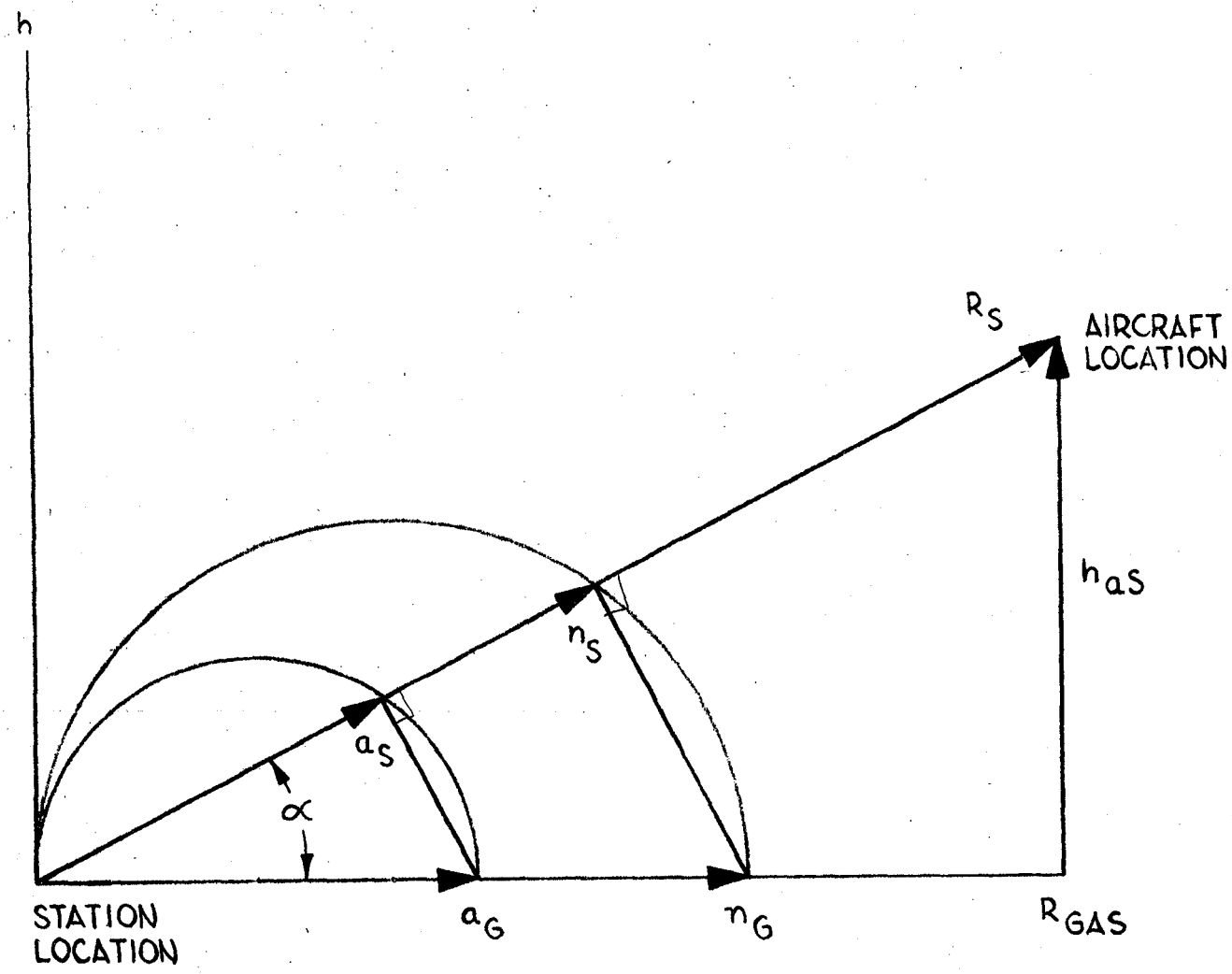


Figure 4-30. A-N Range-Vertical Plane

Since "a" and "n" are the audio signal levels at the aircraft

$$a = \frac{Kas}{R_S} \qquad (2)$$

$$n = \frac{K_{ns}}{R_S} \qquad (3)$$

where K is a constant of proportionality which is dependent on the power and range of the facility

R_S is slant range

a_S and n_S are relative slant range vectors illustrated in figure 4-30.

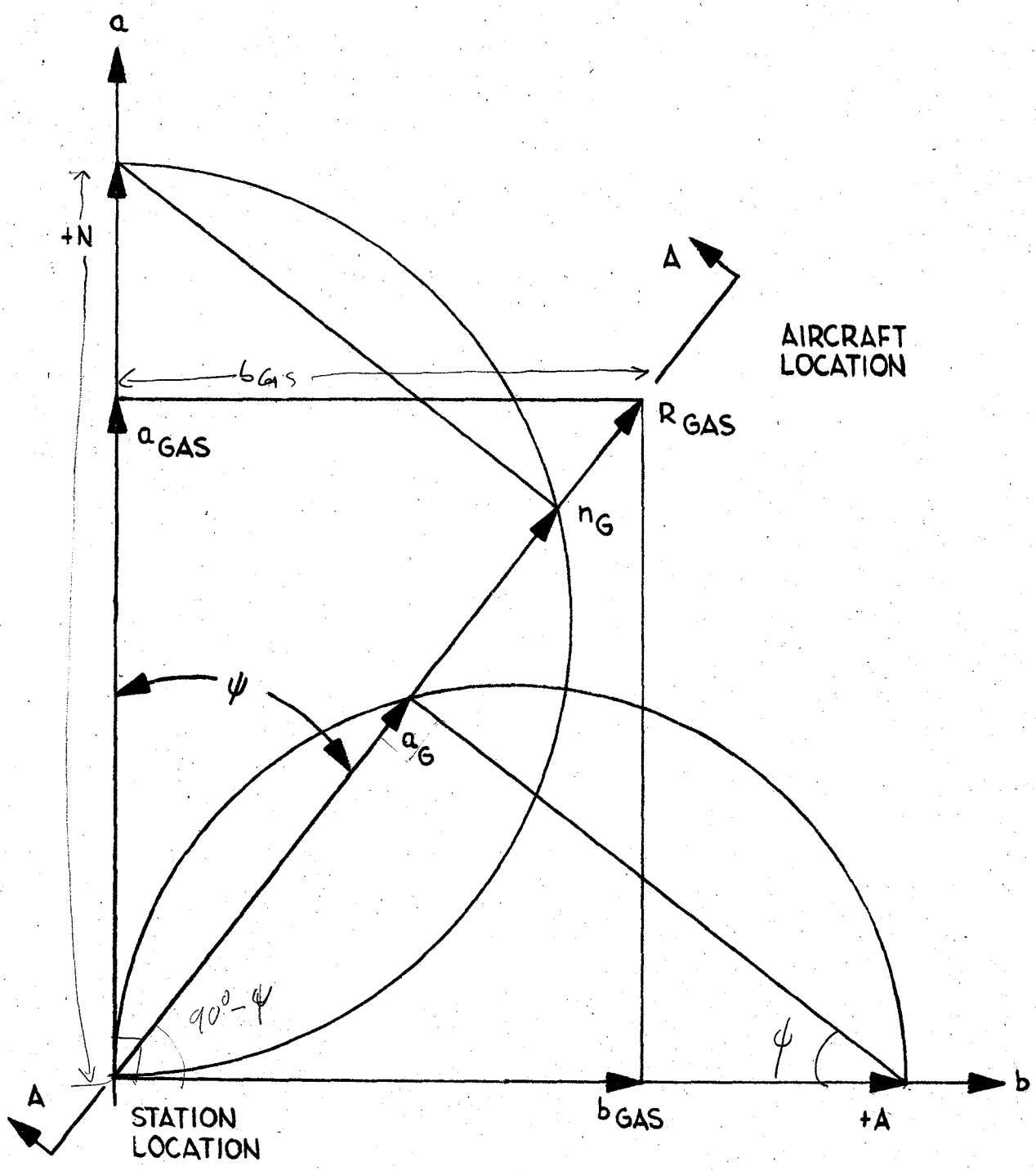


Figure 4-31. A-N Range - Horizontal Plane

From figure 4-30

$$a_S = a_G \cos \alpha \qquad (4)$$

$$n_S = n_G \cos \alpha \qquad (5)$$

$$\cos \alpha = \frac{R_{GAS}}{R_S} \qquad (6)$$

From figure 4-31

$$a_G = A \sin \psi \qquad (7)$$

$$n_G = N \cos \psi \qquad (8)$$

$$\sin \psi = \frac{b_{GAS}}{R_{GAS}} \qquad (9)$$

$$\cos \psi = \frac{a_{GAS}}{R_{GAS}} \qquad (10)$$

Combining equations (2), (4), (6), (7), and (9)

$$a = \frac{KA\ b_{GAS}}{R_S^2} \qquad (11)$$

Similarly, equations (3), (5), (6), (8), and (10) become

$$n = \frac{KN\ a_{GAS}}{R_S^2} \qquad (12)$$

Substituting equations (11) and (12) into equation (1), the servo input becomes

$$\frac{KN\ a_{GAS}}{R_S^2} - \frac{KA\ b_{GAS}}{R_S^2}$$

$$d = \frac{KN\ a_{GAS}}{R_S^2} + \frac{KA\ b_{GAS}}{R_S^2}$$

$$d = \frac{N\ a_{GAS} - A\ b_{GAS}}{N\ a_{GAS} + A\ b_{GAS}} \qquad (13)$$

4-91. The diameters of the four circular patterns required on the "Radio Facility Data" sheet are obtained by using the following equations:

$$\theta = \gamma_3 - \gamma_4 + 180^\circ$$

$$\phi = \gamma_1 - \gamma_2 + 180^o$$

$$\alpha = \gamma_3 - \gamma_2$$

$$\beta = \gamma_1 - \gamma_4 + 360^o$$

where γ_1 through γ_4 are given

$$k = \tan(\beta - \theta) + \left[\frac{1}{\tan\alpha \tan\theta}\right][\tan\alpha + \tan\theta + \tan(\beta - \theta)]$$

$$\tan b = \frac{-k + \sqrt{k^2 + 4}}{2}$$

$$\epsilon^o = \gamma_{3o} - b^o - 180^o$$

where ϵ^o is the angle of rotation of the pattern

$$a = \theta - b$$

$$c = \beta - a$$

$$d = \alpha - b$$

$$D = 1\sqrt{\left(\frac{\tan b}{\tan d}\right)^2 + \tan^2 a + \tan^2 b + 1}$$

$$A = /D \tan a/$$

$$C = /D \tan b/$$

$$B = /C \tan d/$$

where A, B, C and D are the diameters of the four circular patterns.

4-92. A sample "A-N Range Data Sheet" is shown in figure 4-32.

4-93. The A-N Range pattern for the information contained in row 1 of figure 4-32 is shown in figure 4-33.

GIVEN ANGLES								ANGLE OF ROTATION		CIRCLE DIAMETERS							
$\gamma_1 \times 10^6$		$\gamma_2 \times 10^6$		$\gamma_3 \times 10^6$		$\gamma_4 \times 10^6$		$\epsilon \pm 10{,}000$		A ×100		B ×100		C ×100		D ×100	
1 1000	090.0	1000	175.0	1000	229.0	1000	346.0	10	029.225	10	0.6251	10	0.34814	10	0.23709	10	0.66056
2 1000	049.0	1000	156.0	1000	229.0	1000	336.0	10	012.499	10	0.42010	10	0.56841	10	0.42060	10	0.56841
5 1000	057.0	1000	107.0	1000	237.0	1000	302.0	-10	004.566	10	0.51847	10	0.28073	10	0.71026	10	0.38457
13 1000	002.0	1000	092.0	1000	182.0	1000	272.0	-10	043.000	10	0.50000	10	0.50000	10	0.50000	10	0.50000
25 1000	049.0	1000	148.0	1000	229.0	1000	310.0	10	004.000	10	0.57206	10	0.57206	10	0.41562	10	0.41562
31 1000	041.0	1000	123.0	1000	221.0	1000	303.0	-10	008.000	10	0.53366	10	0.46390	10	0.53366	10	0.46390

Figure 4-32. A-N Range Data Sheet

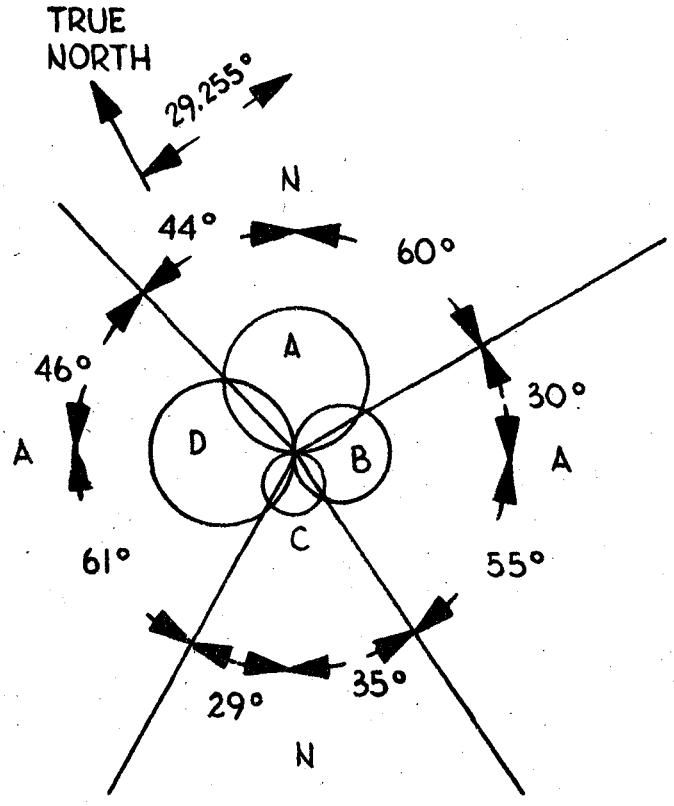


Figure 4-33. A-N Pattern for Row 1 Figure 4-32

4-94. SAWTOOTH GENERATION FOR CONTINUOUS ROTATION SERVO DRIVE.

4-95. General. In order to drive the continuous-rotation servos associated with the simulator peripheral equipment, it is necessary to generate two voltages linear with respect to the shaft angle as shown in figure 4-34.

4-96. If the angle exists explicitly in core memory, there will be a linear function of the angle. This function may have to be modified in some way to get it in a form similar to the "Red" sawtooth of figure 4-34 and, then have unity added to or subtracted from it to get the "Black" sawtooth. This is a matter of routine programming.

4-97. If, however, the sine and cosine of the angle is in core memory the problem is more complex. The starting point is a truncated series for the arcsine:

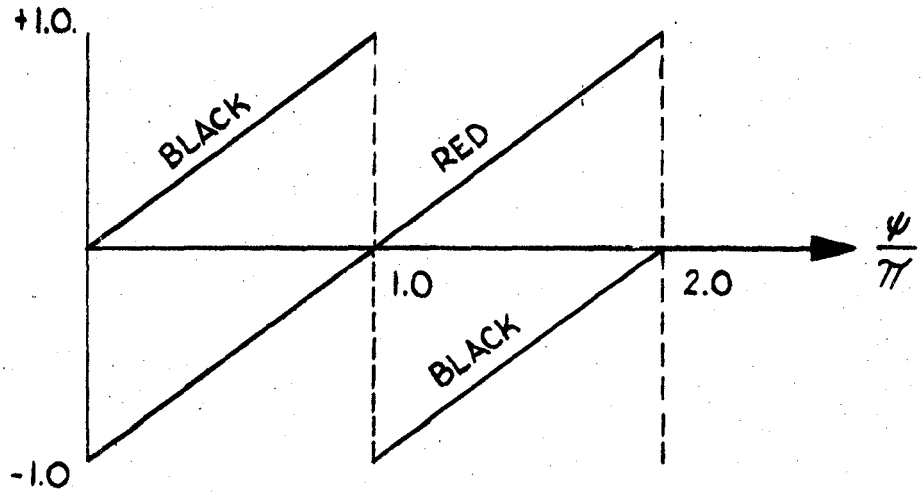


Figure 4-34. Sawtooth Functions

$$\theta = \sin^{-1} Z = Z \left[a' + b' \cdot Z^2\right] \qquad (1)$$

where $a' = 0.986706$
$b' = 0.24803$

4-98. Equation (1), with the given value of the constants, is valid for $-45^\circ \leq 0 \leq +45^\circ$. It is exact at zero and at the end points and has a maximum error of 7.32 minutes in this range.

4-99. Relationship Between θ and ψ. Because of the restricted range of validity of equation (1), the quadrant in which ψ lies must be determined to operate on it appropriately. If the angle lies in the first quadrant, subtraction of 45° guarantees a value for θ within the valid range. Likewise, for a second quadrant angle subtract 135°; third quadrant 225°, and fourth quadrant 315°. Following this scheme the following equations were derived:

First quadrant:
$$Z = \sin\theta - \sin(\psi - 45^\circ)$$
$$= \sin\psi \cos 45^\circ = \cos\psi \sin 45^\circ$$
$$= \frac{1}{\sqrt{2}} \left[\sin\psi - \cos\psi\right] \qquad (2)$$

Second quadrant
$$Z = \theta = \sin(\psi - 135^\circ)$$
$$= \sin\psi \cos 135^\circ - \cos\psi \sin 135^\circ$$
$$= \frac{1}{\sqrt{2}} \left[\sin\psi + \cos\psi\right] \qquad (3)$$

Third quadrant:
$$Z = \sin\theta = \sin(\psi - 225^\circ)$$
$$= \sin\psi \cos 225^\circ - \cos\psi \sin 225^\circ$$
$$= \frac{1}{\sqrt{2}} \left[\sin\psi - \cos\psi\right] \qquad (4)$$

Fourth quadrant:
$$Z = \theta = \sin(\psi - 315^\circ)$$
$$= \sin\psi \cos 315^\circ - \cos\psi \sin 315^\circ$$
$$= \frac{1}{\sqrt{2}} (\sin\psi + \cos\psi) \qquad (5)$$

4-100. Equations (2) through (5) all have the same form, differing only in their prefix signs which are systematic, alternating from quadrant to quadrant. To simplify the equations the following identities are assumed:

$$x = \sin\psi;\ y = \cos\psi;\ \text{and}\ H = /x/ = /y/$$

Equation (1) becomes:

$$\theta = \frac{+H}{2}\left[a' + \frac{b' H^2}{2}\right] \qquad (1a)$$

or, normalizing with respect to π

$$\theta' = \pm H\left[a' + bH^2\right] \qquad (6)$$

where: $a = \frac{a'}{\sqrt{2\pi}} = 0.22208686$

$$b = \frac{b'}{2\sqrt{2\pi}} = 0.02791314$$

The plus sign applies in quadrants I and III, and the minus sign in quadrants II and IV.

4-101. The relationship required between θ and ψ, as defined in paragraph 4-99 and as given by equation (6), in normalized coordinates, is shown in figure 4-35.

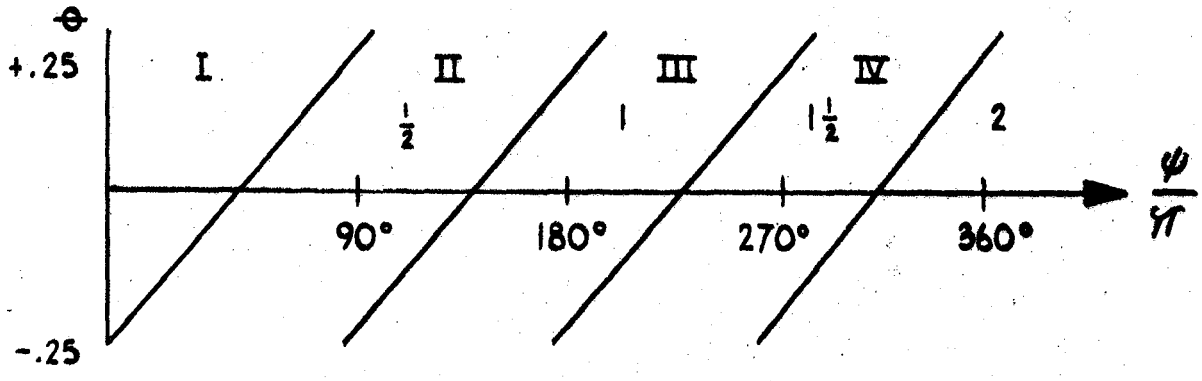


Figure 4-35. Relationship Between θ and ψ In Normalized Ordinates

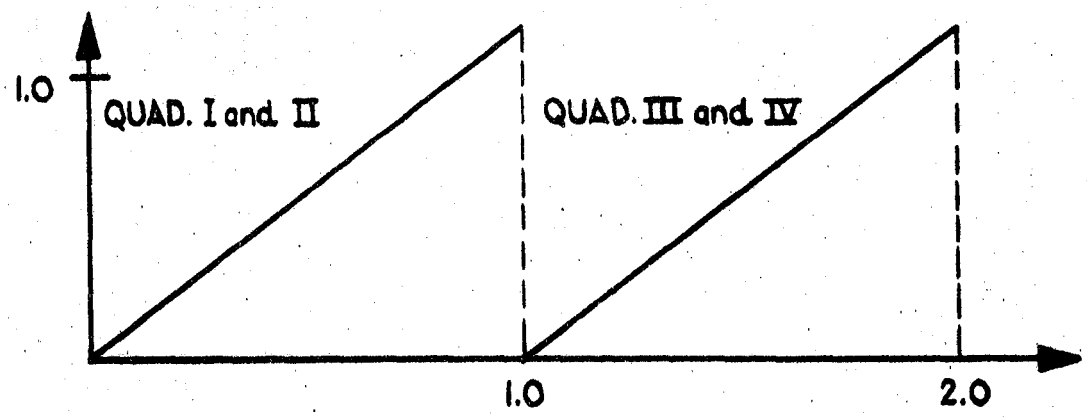


Figure 4-36. Results of Addition to Figure 4-35

4-102. Comparison of figures 4-34 and 4-35 illustrates what must be done in programming to compute the sawtooth. If one-fourth is added to the curves in figure 4-35 corresponding to quadrants I and III, and three-fourths to the curves for quadrants II and IV the result will be as shown in figure 4-36.

4-103. If the angle lies in quadrant I or quadrant II equation (6) has effectively given the "Black" sawtooth. If the angle is in quadrant III or IV solve for the "Red" sawtooth. In either case it is only necessary to subtract unity to obtain the other function.

4-104. Quadrant detection is achieved by defining three Boolean functions:

1. $K_1 = 1$ when $\sin \psi < 0$

2. $K_2 = 1$ when $\cos \psi < 0$

3. $K_3 = K_1 \oplus K_2 = K_1\bar{K}_2 + \bar{K}_1K_2 = 1$ when $K_1 = K_2$

   or $K_3 = \overline{K_1 \oplus K_2} = \bar{K}_1\bar{K}_2 + K_1K_2 = 1$ when $K_1 = K_2$

4-105. $K_1 = 1$ means that $\pi \psi < 2\pi$ and $K_3 = 1$. In the second sense it means that the angle is either in quadrant I or quadrant III. The second form for K_3 is more convenient in implementing the program. These two functions are sufficient to establish all necessary conditions.

4-106. Figure 4-37 is a sample problem of this nature. The "Black" sawtooth is assumed to be stored in core memory location 0646, and the "Red" sawtooth in 0647. Other locations, while given, were chosen at random.

DATE 26 MAY 1963
PROBLEM 360° SERVO DRIVE
PROGRAMMER R. E. MACK

CODING AND CONSTANT SHEET
MARK I COMPUTER

F-2214-A

PAGE 1 OF 2

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
0 1	LD	2 0	2 4 1 2	2^0	$y = \cos \psi$							
0 2	FLN	1 6	0 0 0 0				k2					k2 = 1 In Quadrants II and III
0 3	ABS	1 4	0 0 0 0		$y = /\cos \psi/$							
0 4	LD	2 0	2 4 1 1		$x = \sin \psi$							
0 5	FLN	1 6	0 0 0 0				k1					k1 = 1 In Quadrants III and IV
0 6	ABS	1 4	0 0 0 0		$x = /\sin \psi/$							
0 7	SUB	0 2	0 0 0 0		$/x/ - /y/ \triangleq H$							
0 8	ST	2 3	3 7 6 6	2^0	H							
0 9	SQ	0 5	0 0 0 0									
1 0	BLD	3 3	0 0 0 0				k2					
1 1	AND	3 1	0 0 0 0				k1 k2					
1 2	BLD	3 3	0 0 0 0				k_1					
1 3	OR	3 0	0 0 0 2		H^2		k1 + k2					
1 4	MLT	0 3	2 4 6 0									
1 5	BIN	3 2	0 0 0 0				$\overline{k1}$ $\overline{k2}$					
1 6	OR	3 0	0 0 0 0				$\overline{k3} = \overline{k1} \oplus \overline{k2} = k1\ k2 + \overline{k1}\ \overline{k2}$					$\overline{k3}$ = 1 In Quadrants I and III
1 7	NPA	1 2	0 0 0 0									
1 8	NPB	2 5	3 7 7 7		$b_H 2$							
1 9	ADD	0 1	2 4 6 1		$a + b H^2$							
2 0	MLT	0 3	3 7 6 0									
2 1	NPA	1 2	0 0 0 0									
2 2	NPB	2 5	3 7 7 7									
2 3	NPA	1 2	0 0 0 0									
2 4	NPB	2 5	3 7 7 7		$\ominus = H (a + b H^2)$							
2 5	SKP	2 6	0 0 0 2									Skip If In Quadrants I or III
2 6	INS	1 3	0 0 0 0		$\ominus = H (a + b H^2)$							
2 7	ADD	0 1	2 4 6 2		$\ominus' + 1/2$							
2 8	ADD	0 1	2 4 6 3		$\ominus' + 1/4$ or $\ominus' + 3/4$							Dependent on Conditional Skip
2 9	BLD	3 3	0 0 0 1				k_1					$k_1 = 1$ If Red Sawtooth Has Been Computed
3 0	BST	2 4	0 6 4 6									Either Red or Black

MARK I

Figure 4-37. 360° Servo Drive (Sheet 1 of 2)

DATE 26 MAY 1963

PROBLEM 360° SERVO DRIVE

PROGRAMMER R. E. MACK

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-2214-A

INSTRUCTION NUMBER (1 2 3 4 5 6 7)	MNEMONIC CODE	O.P. CODE (8 9 10)	MEM. ADD. (12 13 14 15)	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
31	SUB	02	2464									Subtract One From the Contents of the Accumulator to Form the Remaining Sawtooth
32	BIN	32	0000				EI					
33	IST	24	0646									
34	LD	20	0647		"Red" Sawtooth							
35	SCL	10	1002		1/4 ("Red" Sawtooth)							Take the "Red" Sawtooth Which Varies From -1 to +1 Over 360°, Divide by Four and Add One-Fourth. The "Red" Sawtooth Will Now Vary From -1/2 to +1/2
36	ADD	01	2463		1/4 ("Red" Sawtooth) + 1/4							
37	ST	23	1000									
												The Above Program Assumes That Steps 34 Through 37 Have Been Completed Previously

MARK I

Figure 4-37. 360° Servo Drive (Sheet 2 of 2)

4-107. RATE DRIVE FOR CONTINUOUS ROTATION SERVOS.

4-108. General. Functions such as ADF manual loop drive or gyro precession terms require a method of driving position servos in such a way as to give the effect of velocity servo performance. The sample program discussed in the following paragraphs gives this effect.

4-109. Two assumptions are to be made in this sample program. The first is that a modified form of the "Red" sawtooth (refer to paragraphs 4-94 through 4-106), associated with the angular function involved (see figure 4-38), has been generated and stored in the machine as part of the basic sawtooth-generation program (paragraph 4-106).

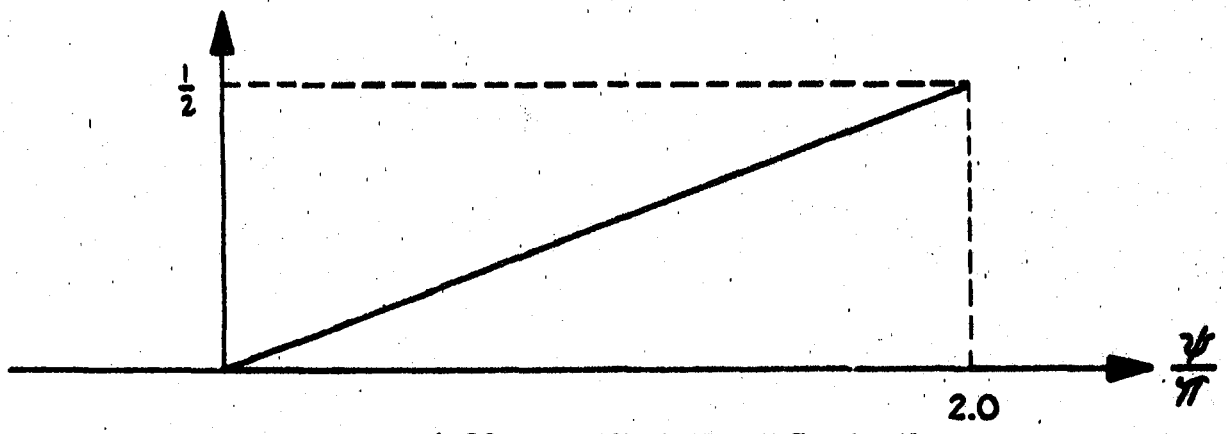


Figure 4-38. Modified "Red" Sawtooth

4-110. The second assumption is that an incremental angle, $\Delta\psi$, as a function of the required rate and the band on which the program has been written has also been stored in the Mark I.

4-111. The modified "Red" sawtooth as shown in figure 4-38 is used for two reasons. First, it makes the program overflow-proof, and, second, it simplifies the generation of Boolean control functions. Let the following functions hold true.

Arithmetic:

$\Delta\psi$ = Incremental Angle

ψR = Last computed value of modified sawtooth

$\psi' R = \psi R + \Delta\psi$ = New value of modified sawtooth

Boolean:

A = 1 when $\Delta\psi$ is negative

B = 1 when $(\psi' R - 1/2)$ is negative

C = 1 when $\psi' R$ is negative

4-112. Examination of figure 4-38 shows that there are two critical points for $\psi'R$ to be checked. Obviously, if $\psi' R$ has a value near zero and $\Delta\psi$ is negative, $\psi'R$ must be checked to see if it is negative. If $\psi' R$ is negative add one-half to obtain the correct value. If $\psi' R$ has a value near one-half and $\Delta\psi$ is positive, $\psi'R$ must be checked to see if it is greater than one-half, If this condition exists, subtract one-half to obtain the correct value. Under any other circumstances, the computed value of $\psi' R$ is the correct value.

4-113. $\psi' R$ must be "operated" on to retrieve the correct form for the "Red" sawtooth, and either add or subtract one, as required, to obtain the "Black" sawtooth. Figure 4-39 is a sample program for a problem of this type.

4-114. DIAGNOSTIC PROGRAM.

4-115. Diagnostic programs serve as a check to ensure that the Mark I is performing functions within its capabilities (adding, subtracting, multiplying etc.).

4-116. Diagnostic programs utilize the "Conditional Stop" instruction in conjunction with the "Conditional Skip" instruction in all phases of the program (i.e., if a certain condition is met, skip to the next part of the program; if not, stop). As readily seen this type of program is a valuable aid to maintenance personnel.

4-117. A sample diagnostic program to check the multiplication process within the Mark I is illustrated in figure 4-40.

4-118. PUNCHED CARDS.

4-119. General. IBM type 5081 punched paper cards are used to contain the permanent program for the Mark I. Each card has 80 vertical columns and ten horizontal rows numbered zero through nine. Above row zero there is space for two punches. The "eleven" punch is parallel with and directly above row zero, with the "twelve" punch in parallel and above the "eleven" punch.

4-120. The allowable characters to be punched on the cards for use in conjunction with the Mark I and Tape Preparation Unit (including Card Reader), are the ten numerics (zero through nine), all upper case alphabetics, and the following special characters:

a. Left Parenthesis (Punch in rows 0, 4, and 8)

b. Right Parenthesis (Punch in rows 12, 4, and 8)

c. Left Bracket (Punch in rows 11, 3, and 8)

d. Right Bracket (Punch in rows 0, 3, and 8)

e. Equal Sign (Punch in rows 3 and 8)

DATE 26 MAY 1963

PROBLEM VELOCITY SERVO DRIVE

PROGRAMMER J. HOOP

F-2214-A

CODING AND CONSTANT SHEET

MARK I COMPUTER

PAGE 1 OF 1

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	LD	20	2763		$\Delta\psi$							Load Incremental Angle
02	FLN	16	0000				A					A = 1 => $\Delta\psi < 0$
03	ADD	01	1000		$\Delta\psi + \psi'R = \psi'R'$							Form $\psi'R$
04	FLN	16	0000				C					C = 1 = $\psi'R$ 0
05	SUB	02	2111		$\psi'R - 1/2$							
06	FLN	16	0000				B					B = 1 => $\psi'R - 1/2 < 0$
07	OR	30	0000				A + B					
08	BIN	32	0000				$\overline{A+B} = \overline{A}\,\overline{B}$					
09	SKP	26	0006									
10	ADD	01	2111		$\psi'R$							Add 1/2 to Recover $\psi'R$
11	BLD	33	0001				A					
12	AND	31	0001				AC					
13	BIN	32	0000				$\overline{AC} = \overline{A} + \overline{C}$					
14	SKP	26	0001									
15	ADD	01	2111		$\psi'R + 1/2$							
16	ST	23	2000		ψR							
17	SUB	02	1005		$\psi R - 1/4$							
18	SCL	10	1002		$4\ [\psi R - 1/4]$							
19	ST	23	2001									Store "Red" Sawtooth
20	FLN	16	0000				D					D = 1 => $\psi R - 1/4 < 0$
21	SKP	26	0003									
22	SUB	02	2500		$4\ [\psi R - 1/4] - 1$							Subtract One to Get "Black" Sawtooth
23	BIN	32	0000				$\overline{D}$					
24	SKP	26	0001									
25	ADD	01	2500		$4\ [\psi R - 1/4] + 1$							Add One to Get "Black" Sawtooth
26	ST	23			$4\ [\psi R - 1/4) + 1$ or $4\{\psi R - 1/4] - 1$							Store "Black" Sawtooth

MARK I

Figure 4-39. Velocity Servo Drive Program

DATE 6-27-63

PROBLEM DIAGNOSTIC TEST (MULTIPLY)

PROGRAMMER S. JONES

CODING AND CONSTANT SHEET

MARK I COMPUTER

F-2214-A

PAGE 1 OF 1

INSTRUCTION NUMBER	MNEMONIC CODE	O.P. CODE	MEM. ADD.	SCALE	ARITHMETIC ACCUMULATOR	SALV.	BOOLEAN ACCUMULATOR	SALV. 1	SALV. 2	SALV. 3	SALV. 4	REMARKS
01	LDK	22	5252		10101010101							Load Constant
02	SCL	10	1005									Scale Right Five Places
03	SCL	10	1005									Scale Right Five Places
04	SCL	10	1002									Scale Right Two Places
05	LDK	22	5252		10101010101							Load Constant
06	ADD	01	6000		10101010101010101010101 = N = 2/3							Add Salvage Register to Accumulator to Form Constant N = 2/3
07	LDK	22	2000		2^{-2} = (1/4)		lost					Load Constant
08	MLT	03	0000		2/3 (2^{-2})							Multiply By Salvage
09	NPA	12	0000									
10	NPB	25	3777									
11	NPA	12	0000									
12	NPB	25	3777		2/12 = 1/6							
13	SFT	11	0002		1/6 (4) = 2/3 = A							Shift Two Places To Form Constant A = 2/3
14	SUB	02	0000									Subtract Salvage (A - N = Error)
15	SCL	10	0005									
16	SCL	10	0005									
17	SCL	10	0005									
18	SCL	10	0005									
19	SCL	10	0002									
20	SFT	11	0001		(+)							Place The Last Digit Of Answer In The Sign Location
21	INV	13	0000		(-)							
22	FLN	16	0000				1					
23	SKP	26	0001									Skip To Next Phase Of Program If Answer Correct (Equals 0)
24	STP	27	0000									Stop If Answer Incorrect

MARK I

Figure 4-40. Diagnostic Program

f. Plus Sign (Punch in rows 4 and 8)

g. Minus Sign (Punch in row 11)

h. Decimal Point (period) (Punch in rows 12, 3, and 8)

i. Asterisk (Punch in rows 11, 4, and 8)

j. Division Sign (Punch in rows 0 and 1)

k. Ampersand or Plus Sign (Punch in row 12)*

*Used to arm the TPU

NOTE

An "11" punch plus one of the allowable numerics punched in the same column is used for the flag bit. A "12" punch and "11" punch in the same column constitutes the Block End Code.

4-121. Figure 4-41 shows all allowable characters that may be punched on a card when used with the Mark I and Tape Preparation Unit.

4-122. The first card for the first quadrant of each General Purpose or Interpolator band must contain a "12" punch followed by seven NPA instructions. For all other General Purpose or Interpolator band quadrants, and the four Data Preselector band quadrants, the first card must contain a "12" punch followed by one NPA instruction.

4-123. General, Interpolator, Preselector and Core Memory programs have individual card formats.

4-124. General Program Card Formats. General Programs may have either of two card formats:

a. One instruction per card (same as the "Coding and Constant" sheets).

b. Eight instructions per card plus a sort number, and an equation "header" card for each individual program within the overall General Program.

4-125. The card format using one instruction per card is as follows:

a. Column 1 through column 4 are reserved for the equation number. The equation number is dependent upon the type of program being prepared. Table 4-2 lists the equation numbers for the various types of General Programs.

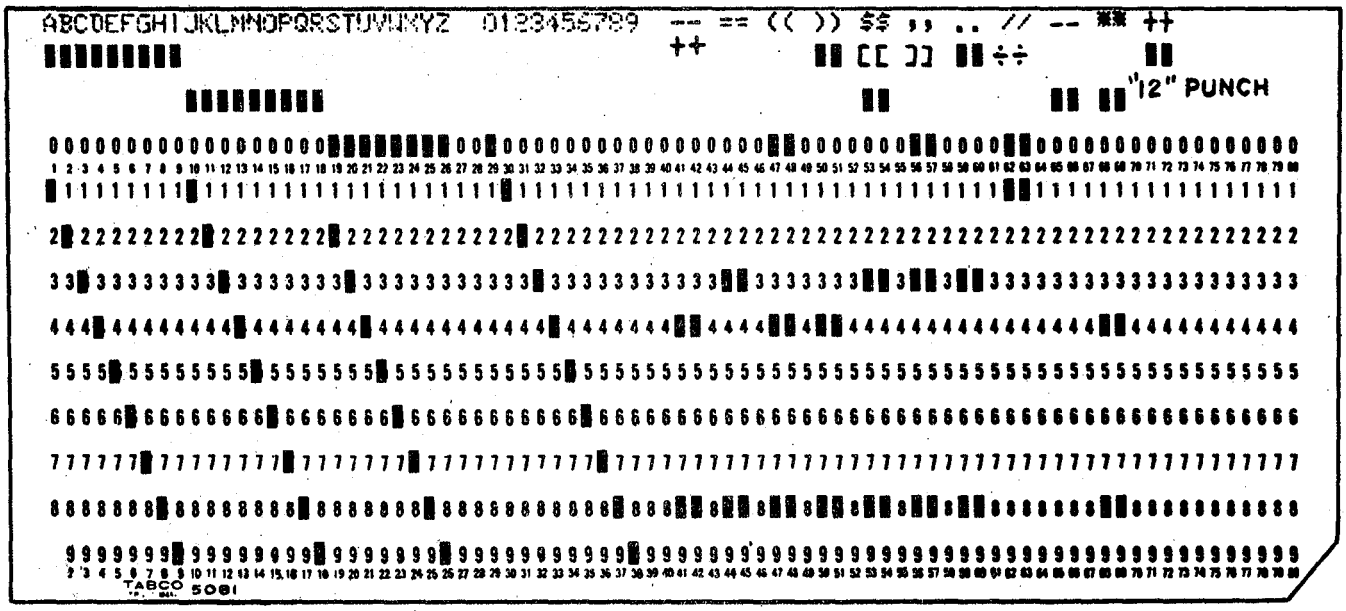


Figure 4-41. IBM Type 5081 Paper Card

Table 4-2. General Program Equation Numbering System

Equation Numbers	Simulator Section
000 - 0999	Flight Equations
1000 - 1999	Expansion
2000 - 2999	Flight Accessories
3000 - 3999	Expansion
4000 - 4999	Engines
5000 - 5250	Air Conditioning, Pressurization etc.
5251 - 5500	Electrical
5501 - 5750	Hydraulics
5751 - 5999	Expansion
6000 - 6999	Engine Accessories
7000 - 7999	Expansion
8000 - 8999	Nav. Comm.
9000 - 9499	Expansion
9500 - 9999	Misc. Simulator

b. Columns 5 through 7 are used for the instruction numbers within the specific program.

c. Column 8 always has a "12" punch to inform the TPU that an instruction is about to be received.

d. Columns 9 and 10 are used for the O.P. Code.

e. Column 11 is left blank.

f. Column 12 through column 15 are reserved for the memory address.

g. Column 16 always has an "A" punch which disables the TPU.

h. Column 17 contains either a plus or minus sign for the scale factor.

i. Columns 18 and 19 are reserved for the scaling factor.

j. Column 20 is left blank.

k. Columns 21 through 80 are reserved for remarks.

4-126. The card formats using an equation "header" card and eight instructions per card are as follows:

a. Equation "header" card.

(1) Columns 1 through 4 contain the equation number. (Refer to paragraph 4-125.)

(2) Columns 5 through 20 are left blank.

(3) Column 21 is either F, M, or S to indicate the type of band to be used.

(4) Column 22 would contain a number one through eight to indicate which one of the particular bands is to be used.

(5) Column 23 is used to indicate which drum quadrant the equation is on. If the equation is for slow band four quadrant three, column 21 would contain S, column 22 would be four and column 23 would be three.

(6) Column 24 is left blank.

(7) Columns 25 through 26 contain the three digits giving the total number of instructions for the equation. (The header card is considered an instruction.) Thus, a 29 step program appears as 030.

(8) Column 27 is left blank.

(9) The trainer designation is printed using columns as required.

(10) The equation name is printed after the trainer designation.

(11) The name or initials of the programmer follows the equation name.

(12) The initials of person making the current revision (where applicable) follow the original programmer's name or initials.

(13) Next the date is printed (this is the original date of the program unless revisions have been made, in which case, the current revision date is used).

(14) Last, the current revision letter is printed (when applicable).

b. Eight instructions per card format.

(1) Column 1 of the first card in the program contains a "12" punch followed by the NPA instruction. All other cards required in the program would have a "12" punch in column 1 followed by the O.P. Code and Memory Address.

(2) Columns 11, 21, 31, 41, 51, 61 and 71 all contain a "12" punch followed by the O.P. Code and Memory Address.

(3) Columns 78 through 80 contain the octal number of the card in the General Program.

4-127. Linear Function Interpolator Card Formats. The card format for all Interpolator programs is consistent except for the number of cards required to complete the program and the sort number. The sort number is a combination of two alphabetics and three numerics starting with AA001. Card formats for one, two and three variable functions follows.

4-128. Single Variable Function. A single variable function will require five cards to complete the program. All cards will have a "12" punch in column 1 followed by an "11" punch plus the start of the first word in column 2. (The "11" punch is used for the flag bit.) There will be an "A" punched in column 73 of all cards in the program, and the sort number punched in columns 76 through 80.

Card 1. The control word will be punched starting with 100 in column 2.

Card 2. The address of the independent variable X punched five times starting in columns 2, 10, 18, 26 and 34, and preceded by a "12" punch.

Card 3. Contains the dependent variables starting in column 2. Each additional dependent variable will be punched starting eight columns from the start of its predecessor. All dependent variables will be preceded by a "12" punch.

Card 4. Contains the computer time starting in columns 2 and 10 and preceded by a "12" punch.

Card 5. Contains the answer address repeated five times. This card will also have an "11" punch in column 34.

4-129. Two Variable Function. A two variable function will require a maximum of 14 cards to complete the program depending upon the size of the data field.

Card 1. Same as Card 1 paragraph 4-128 except the control word will start with 200 in column 2.

Card 2. Same as Card 2 paragraph 4-128.

Card 3. Address of independent variable Y. Card format is the same as Card 2 paragraph 4-128.

Card 4. Same as Card 3 paragraph 4-128.

Card 5- Card 12 (where required). Same as Card 3 paragraph 4-128 except there will not be an 11 punch in column 2 in any of these cards.

Card 13. Computer Time. Same as Card 4 paragraph 4-128 except repeated in columns 18 and 26.

Card 14. Answer Address. Same as Card 5 paragraph 4-128.

4-130. Three Variable Function. A three variable function could require up to 87 cards to complete the program.

Card 1. Same as Card 1 paragraph 4-128 except the control word will start with 300 in column 2.

Card 2 and Card 3. Same as Card 2 and 3 paragraph 4-129.

Card 4. Address of the independent variable Z. Card format is the same as Card 2 paragraph 4-128.

Card 5. Same as Card 4 paragraph 4-129.

Card 6 - Card 85 (where required). Same as Card 5 - Card 12 paragraph 4-129.

Card 86. Same as Card 13 paragraph 4-129 except repeated in columns 34 and 42.

Card 87. Same as Card 5 paragraph 4-128.

4-131. Data Preselector Card Format. The card format for Data Preselector programs is consistent except for the number of cards required to program the various groups of facilities. (See Appendix A.)

4-132. Each Data Preselector Card will contain the following information:

a. Type of facility

b. Call letters

c. Memory address . (Refer to paragraph 3-64.)

d. Simulator designation

e. Facility information (one word per card i.e., latitude, longitude, frequency etc.).

The binary equivalent of the facility information may also be punched on the card, but this is optional.

4-133. A "12" punch used to inform the TPU that information is about to be received must be in the column directly preceding the facility information.

4-134. Data Preselector cards will have the first letter of the facility type punched in column 1, and the last numeric of the facility information (binary equivalent only) in column 80. All other information is approximately evenly spaced across the card.

4-135. Core Memory Cards. The punched paper cards used in conjunction with the Core Memory have no specific card format. The only requirement is that a "12" punch appear in the column directly preceding the Core Memory word. The Core Memory card usually contains from one to four Core Memory words.

4-136. TAPE FORMATS.

4-137. Work Tapes. A paper work tape shall be prepared on the Tape Preparation Unit (TPU) through keyboard entry or by using the tape reader to copy another work tape. Work tapes shall have a maximum content of one 1,024-word block of data, and shall not be used to operate the drum loader. Work tapes can be reproduced with blank spaces but shall ignore deletes.

4-138. Work Tape Special Codes. The following special codes shall be included in a work tape:

a. Block End, or Stop Code: 11111·001 shall indicate the end point of a quadrant (block of 1,024 words).

b. Delete Code: 11111:111 shall be used by the operator to delete any desired tape data word. This code must be punched over all lines of the work to be deleted.

c. Gap Code: 00000:000 shall be used as desired between data words to provide tape gaps for later insertion of modified words or for tape splicing. A work tape may be spliced between any two data words.

d. Dummy Code: 0111 shall be placed at the most signification end of the first line of each data word except for general purpose instruction words. This dummy code will be punched automatically in each first line when used.

4-139. Master Tapes. A master tape (paper backed Mylar) shall be prepared on the TPU by using: (1) the tape reader to copy a work tape or a previous master tape; (2) keyboard entry, or (3) readout of the accumulator or core memory contents.

4-140. The form of the data words on a master tape shall be identical with their form on a work tape, but delete codes shall not be permitted on the master, and there will be no gap codes, except between data blocks. The transformation of a work tape to a master tape shall consist of removal of all deleted data words and of all gaps within the data block.

4-141. If the master tape contains more than one data block, the succeeding blocks must be arranged in block address order. The setting of the first block address into the drum loader is performed manually, but succeeding blocks shall be transferred automatically to addresses in ascending order. Table 4-3 lists the address for the first word in each block (quadrant), and the band on which the quadrant is located.

4-142. The final block of a multiple-block master tape will have a block end code (paragraph 4-138) modified to a reel end code: 11111.101, through the operation of a manual keyboard control. In a single-block master tape, the block end code will stop the drum loader, but in a multiple-block master, a control change will cause the block end code only to load the block data and then continue, while the reel end code stops the drum loader.

4-143. Each tape used for loading the first quadrant of every General Purpose or Interpolator band must be started with seven NPA instructions while every other quadrant associated with either the General Purpose or Interpolator bands must be started with one NPA instruction. Data Preselector tape need only have one NPA instruction for the start of each quadrant.

Table 4-3. Drum Loader Quadrant Addresses

BAND	QUADRANT	ADDRESS
FAST	1	000
	2	001
	3	002
	4	003
MEDIUM 1	1	010
	2	011
	3	012
	4	013
MEDIUM 2	1	020
	2	021
	3	022
	4	023
MEDIUM 3	1	030
	2	031
	3	032
	4	033
SLOW 1	1	040
	2	041
	3	042
	4	043
SLOW 2	1	050
	2	051
	3	052
	4	053
SLOW 3	1	060
	2	061
	3	062
	4	063
SLOW 4	1	070
	2	071
	3	072
	4	073

Table 4-3. Drum Loader Quadrant Addresses (Cont)

BAND	QUADRANT	ADDRESS
INTERPOLATOR 1	1	130
	2	131
	3	132
	4	133
INTERPOLATOR 2	1	140
	2	141
	3	142
	4	143
INTERPOLATOR 3	1	150
	2	151
	3	152
	4	153
INTERPOLATOR 4	1	160
	2	161
	3	162
	4	163
DATA PRESELECTOR	1	170
	2	171
	3	172
	4	173

4-144. PREPARATION OF DUPLICATE TAPE (CHANGE IN ONE OR MORE COMPUTER WORDS).

4-145. To prepare a new tape with corrections or changes in certain address locations (tape to tape), proceed as follows:

a. Set the four location digiswitch (located on the tape reader panel) to the first address where a correction is to occur.

b. Activate the proper mode button for the type of tape to be changed (i.e., General Purpose, Interpolator, Data Preselector etc.).

c. Place the Tape Preparation Unit in the tape reader mode by activating either the "Master Tape" or "Work Tape" buttons.

d. Press the "select" button, then the "start" button both located below the digiswitch.

e. When the tape reader stops, activate the keyboard select switch and enter the new computer word or words via the keyboard.

f. Activate the "tape reader" button.

g. Activate the "continue" button.

h. Activate the "start" button.

NOTE

The master clear button must be activated when changing modes.

MARK I

APPENDIX A

MARK I COMPUTER MNEMONIC AND NUMERIC CODES

Operation	Mnemonic Code	Numeric Code
Add	ADD	01
Subtract	SUB	02
Multiply	MLT	03
Negative Multiply	NMT	04
Square	SQ	05
Divide	DIV	06
Square Root Step	SRS	07
Scale	SCL	10
Shift	SFT	11
No-Operation	NPA	12
Invert Sign	INS	13
Absolute Value	ABS	14
Zero Slice	ZSL	15
Flag Negative	FLN	16
Load Accumulator	LD	20
Index Load Accumulator	ILD	21
Load Constant	LDK	22
Store Accumulator	ST	23
Index Store Accumulator	IST	24
No-Operation	NPB	25
Conditional Skip	SKP	26
Conditional Stop	STP	27
Boolean Sum	OR	30
Boolean Product	AND	31
Invert Boolean Accumulator	BIN	32
Load Boolean Accumulator	BLD	33
Store Boolean Accumulator	BST	34
Tape Stop Code	TRS	37

MARK I

TABLES OF POWERS OF 2

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

RADIO AIDS TYPE CODES

Type Code 1 -	MM
Type Code 2 -	OM
Type Code 3 -	FM
	LFM
	BONE
	Z
Type Code 4 -	ILS
	ILSDME
Type Code 5 -	LVR
	MVR
	HVR
	LVTDME
	MVRDME
	HVRDME
	LVRLTC
	LVRMTC
	LVRHTC
	MVRLTC
	MVRMTC
	MVRHTC
	HVRLTC
	HVRMTC
	HVRHTC
	LTC
	MTC
	HTC
	UHF DF
Type Code 6 -	H
	HH
	MH
	BC
	LMM
	LOM
Type Code 7 -	MA
	MAZ
	ML
	MLZ
	MRA
	MRAZ
	MRL
	MRLZ
	RA
	RAZ
	RL
	RLZ

MARK I

RADIO PRESELECTOR TYPE CODE WORDS

Type Code 1 or 2 MM or OM

Card 1 - Word 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 2 - Word 2 * 0 0 0 0 0 0 0 0 0 * 1 1 1 1 1 1 1 1 1
Card 3 - Word 3 Latitude lower (10) Latitude upper (10)
Card 4 - Word 4 Longitude lower (10) Longitude upper (10)
Card 5 - Word 5 Sign (1) Latitude (19)
Card 6 - Word 6 Sign (1) Longitude (19)
Card 7 - Word 7 sine of axis (10) cosine of axis (10)
Card 8 - Word 8 MISC (6) 0 0 0 0 elevation +1000 (10)

Type Code 3 FM, Z, BONE, LFM

Card 1 - Word 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 2 - Word 2 * 0 0 0 0 0 0 0 0 0 * 1 1 1 1 1 1 1 1 1
Card 3 - Word 3 Latitude lower (10) Latitude upper (10)
Card 4 - Word 4 Longitude lower (10) Longitude upper (10)
Card 5 - Word 5 Sign (1) Latitude (19)
Card 6 - Word 6 Sign (1) Longitude (19)
Card 7 - Word 7 sine of axis (10) cosine of axis (10)
Card 8 - Word 8 type (4) 0 0 0 0 0 0 elevation +1000 (10)
Card 9 - Word 9 call letters (16) MISC (4) (16 one's for Z)
Card 10 - Word 10 call letters (16) MISC (2) 00 (16 one's for Z)
Card 11 - Word 11 call letters (16) 0 0 0 0 (16 one's for Z)
Card 12 - Word 12 call letters (16) 0 0 0 0 (16 one's for Z)

Type Code 4 ILS, ILSDME

Card 1 - Word 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 2 - Word 2 frequency (10) frequency (10)
Card 3 - Word 3 Latitude lower (10) Latitude upper (10)
Card 4 - Word 4 Longitude lower (10) Longitude upper (10)
Card 5 - Word 5 Sign (1) Latitude (19)
Card 6 - Word 6 Sign (1) Longitude (19)
Card 7 - Word 7 Sine of axis (10) cosine of axis (10)
Card 8 - Word 8 1 DME MISC (6) Elevation +1000 (12) (10 for ILS, 11 for ILSDME)
Card 9 - Word 9 Call letter (16) g.p.a. (4) (MSB)
Card 10 - Word 10 Call letters (16) g.p.a. (4) (LSB)
Card 11 - Word 11 Call letters (16) R_L - R_{tg} (4) (MSB)
Card 12 - Word 12 Call letters (16) R_L - R_{tg} (4) (LSB)

Type Code 6 RBN, LOM, LMM, BC

Card 1 - Word 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 2 - Word 2 frequency lower (10) frequency upper (10)
Card 3 - Word 3 Latitude lower (10) Latitude upper (10)
Card 4 - Word 4 Longitude lower (10) Longitude upper (10)
Card 5 - Word 5 Sign (1) Latitude 19
Card 6 - Word 6 Sign (1) Longitude 19
Card 7 - Word 7 0 frequency 12 0 0 0 0 0 0 0
Card 8 - Word 8 0 power (9) elevation +1000 (10)
Card 9 - Word 9 Call letters (16) MISC (4)
Card 10 - Word 10 Call letters (16) MISC (2) 0 0
Card 11 - Word 11 Call letters (16) 0 0 0 0
Card 12 - Word 12 Call letters (16) 0 0 0 0

Type Code 7 LFRR

Card 1 - Word 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 2 - Word 2 frequency lower (10) frequency upper (10)
Card 3 - Word 3 Latitude lower (10) Latitude upper (10)
Card 4 - Word 4 Longitude lower (10) Longitude upper (10)
Card 5 - Word 5 Sign (1) Latitude (19)
Card 6 - Word 6 Sign (1) Longitude (19)
Card 7 - Word 7 0 frequency (12) 0 0 0 0 0 0 0
Card 8 - Word 8 1 power 9 elevation +1000 (10)
Card 9 - Word 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 10 - Word 10 Call letters (16) MISC (4)
Card 11 - Word 11 Call letters (16) MISC (2) 0 0
Card 12 - Word 12 Call letters (16) 0 0 0 0
Card 13 - Word 13 diameter circle 1 (10) diameter circle 2 (10)
Card 14 - Word 14 diameter circle 3 (10) diameter circle 4 (10)
Card 15 - Word 15 sine axis (10) cosine axis (10)
Card 16 - Word 16 Sine mag. var. (10) 0 cosine mag. var. (9)

Type Code 5 VOR, TACAN, UHF/DF

Card 1 - Word 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Card 2 - Word 2 frequency (10) frequency (10)
Card 3 - Word 3 Latitude lower (10) Latitude upper (10)
Card 4 - Word 4 Longitude lower (10) Longitude upper (10)
Card 5 - Word 5 Sign (1) Latitude (19)
Card 6 - Word 6 Sign (1) Longitude (19)
Card 7 - Word 7 sine mag. var. (1) 0 cosine mag. var. (9)
Card 8 - Word 8 0 type (6) 0 0 0 elevation +1000 (10)
Card 9 - Word 9 Call letters (16) MISC (4)
Card 10 - Word 10 Call letters (16) MISC (2) 0 0
Card 11 - Word 11 Call letters (16) 0 0 0 0
Card 12 - Word 12 Call letters (16) 0 0 0 0

* Octal Input	Binary Output
1. 03 (0 to 1/2)	00000000001111111111
2. 27 (1/2 to full)	10000000001111111111
3. 07 (0 to full)	00000000001111111111

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont)

4000 (Octal) to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 (Octal) to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 (Octal)	3072 (Decimal)
to	to
6777 (Octal)	3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 (Octal)	3584 (Decimal)
to	to
7777 (Octal)	4095 (Decimal)

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949