AD742965

(1)

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

(1)

AD742965

AIR FORCE INSTITUTE OF TECHNOLOGY

AIR UNIVERSITY
UNITED STATES AIR FORCE

SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Air Force Institute of Technology (AFIT-EN)
Wright-Patterson AFB, Ohio 45433

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
—

3. REPORT TITLE
Software Simulation of the Minuteman D17B Computer

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
AFIT Thesis

5. AUTHOR(S) (First name, middle initial, last name)
Bruce Chatterton
Captain USAF

6. REPORT DATE
March 1972

7a. TOTAL NO. OF PAGES
155

7b. NO. OF REFS
8

8a. CONTRACT OR GRANT NO.
—

b. PROJECT NO.
N/A

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
GE/EE/72-7

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES
Approved for public release; IAW AFR 190-17
Stephen B. Ingram, Captain, USAF
Director of Information, AFIT

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

A software program has been written which simulates the functions of the Minuteman D17B computer at the register transfer level. The simulation program is written in the FORTRAN Extended language to be used on the Intercom System (teletype) of a CDC 6600 computer system. The simulation program consists of a main program and eight subroutines. A programming language for the D17B simulation was formed which contains numbers and load codes, switches, and miscellaneous commands. Example programs run on the simulated computer have been included to show the types of output available.

DD FORM 1 NOV 65 1473

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer Simulation						
Digital Computer						
Simulation Languages						
FORTRAN Programming						
D17B Computer						

Unclassified
Security Classification

SOFTWARE SIMULATION OF THE MINUTEMAN D17B COMPUTER

THESIS

GE/EE/72-7 Bruce Chatterton
Captain USAF

SOFTWARE SIMULATION
OF THE
MINUTEMAN D17B COMPUTER

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

by

Bruce Chatterton, B.S.E.E.
Captain USAF

Graduate Electrical Engineering

March 1972

Preface

The contents of this thesis represent the results of doing a software simulation of the Minuteman D17B Computer. The D17B computer is a general-purpose computer which was used in the control of the Minuteman Missile. This computer is being phased out of Air Force inventory, and as a result of being declared excess, it is being made available to government agencies and educational institutions. The Air Force Institute of Technology Electrical Engineering Department has acquired two of these computers.

Research has been started at AFIT to make the D17B computer operational in a laboratory environment and to develop applications. The software simulation is a part of this research effort. The other areas being pursued at the present time are the design and construction of a hardware control console, the design and construction of an I/O interface for controlling a tape reader, tape punch, and teletype, and a description of the D17B computer and the steps to be followed in making it operational.

I want to express my appreciation to Dr. Frank M. Brown and Dr. Gary B. Lamont for proposing the simulation program as an area of research and for their expertise as advisors for this project. Special acknowledgement is due Bob Mitchell, a Systems Engineer from Newark Air Force Station, Ohio, for the knowledge and documentation which he has imparted to this research project. I am also grateful to the other four students who were doing research in this area for their help in understanding the operation of the D17B computer.

Bruce Chatterton

Contents

Contents

List of Figures

Abstract

A software program has been written which simulates the functions of the Minuteman D17B computer at the register transfer level. The simulation program is written in the FORTRAN Extended language to be used on the Intercom System (teletype) of a CDC 6600 computer system. The simulation program of the D17B computer was developed at the Air Force Institute of Technology as part of a research effort in making a D17B operational in a laboratory environment. The simulation program has proven itself useful as a teaching aid and can be used for error checking program tapes to be run on the D17B computer. It can also be used as a standard for the hardware version of the computer. The simulation program consists of a main program and eight subroutines. The main program consists of a reading and translation section which reads and interprets input data, a noncompute mode section which implements the loading and interaction functions, and a compute mode section which implements the search, read and write memory, and execute functions. A programming language for the D17B simulation program was formed which contains numbers and load codes, switches, and miscellaneous commands. The miscellaneous commands include such functions as register display, memory display, mode tracing, and setting of flipflops. Example programs run on the simulated computer have been included to show the types of output available.

SOFTWARE SIMULATION OF THE MINUTEMAN D17B COMPUTER

I. Introduction

The purpose of this thesis is to describe the software simulation program of the Minuteman D17B computer that has been developed. A software simulation of the D17B computer was developed as part of a research effort at the Air Force Institute of Technology. This research effort was concerned with finding useful applications for the D17B computer.

There are several reasons why a simulation of the D17B computer was written. This program can be used in teaching the operation of the D17B computer. It can also be used as backup capability for running D17B programs when the actual computer is not available. The most important reason, however, is that the simulation program can provide error checks for the D17B programs which it executes. The hardware version of the D17B computer has no error checking capability.

The simulation program was written to simulate the D17B computer at the register transfer level. A register transfer approach was used because it allowed the D17B computer to be simulated at the information and data transfer level. Thus it was not necessary to simulate the logic equations required to clear and set each flipflop. Using the register transfer approach also allows for the tracing of the information flow in the simulated computer as data is loaded and programs executed. With this information tracing capability, the simulation program can be used as a teaching aid.

General D17B Description. The D17B computer is a small, synchronous, serial, general-purpose digital computer. It was designed to be used in airborne control applications and was used in controlling the guidance and operation of a Minuteman Missile. This computer has several important characteristics of which the following are important to an understanding of the simulation program. (Ref 6:5-6)

1. When the D17B computer is executing, all computer operations are controlled by an internally stored program. This stored program can be entered by external input devices (tape reader, teletype, control console switches, etc.).

2. The word length for this computer is 27 bits, of which 24 are used in computation. The remaining 3 bits are spare and synchronizing bits and thus were not needed in the simulation program. For this reason the word length is treated as 24 bits throughout the remainder of this thesis.

3. The memory storage capability consists of a 6000 rpm magnetic disk with a storage capacity of 2935 words of which 2728 are addressable. The contents of memory include 20 cold-storage channels of 128 sectors (words) each, a hot-storage channel of 128 sectors, four rapid access loops (U,F, E,H,) of 1, 4, 8, and 16 words respectively, four 1-word arithmetic loops (A,L,N,I), and two 4-word input buffer

loops (V,R). Cold-storage channels are those memory locations which allow data to be stored only when they are enabled by an external switch. However, data can be read from them at all times. Hot-storage channels can be used for storing and reading of data without an enable switch. A loop consists of a word or group of words which are continually read and stored on the disk as it turns. A 1-word loop would be read and stored each wordtime. For a 4-word loop, each word is read and stored in four wordtimes, an 8-word loop is read and stored in eight wordtimes. A wordtime is the amount of time required to serially read and store the 24 bits of a word. All portions of memory described here have been included in the simulation program.

4. The D17B computer performs computations using the binary number system with negative numbers being represented in two's complement form (sign plus two's complement).

5. The instruction set for this computer consists of 39 instructions. The mnemonic and octal coding for each instruction is given in Appendix B. Also included with the instruction set is the number of wordtimes required for the execution phase of each instruction.

6. The input capability of the D17B computer includes acceptance of detector, discrete, incremental, and character inputs. The detector input sets the DR (detector reset) flipflop to "1" when a true level is put on the detector input line. Discrete inputs are true or false levels on the discrete input lines. Incremental inputs are sampled

inputs that are incrementally added to the input buffer loops (V,R). Character inputs are five bit codes generated by a teletype or tape reader and transferred to the D17B on the character input lines.

7. The outputs that can be realized from the D17B computer are binary, discrete, single character, phase register status, telemetry, and voltage outputs. Binary outputs are computer generated levels of +1 or -1 available on the binary output lines. A discrete output is a true level which is put on one of 28 discrete-output lines. Only one discrete output line can be at the true level at a time. Single character is a computer generated five bit code of the 4 most significant bits of the accumulator plus a parity bit. The character output is made available on output lines for driving a teletype, a tape punch, or some other character coded output device. Phase register status is the condition of the phase register flipflops which is available for monitoring on output lines. Telemetry output is the bit configuration of registers or voltage signals available on output lines for transmission to telemetry equipment. Voltage output is a computer generated analog voltage corresponding to portions of the accumulator contents which is made available on output lines.

8. Special features of the D17B computer include flag store, split-word arithmetic, and minimized access timing. Flag store provides the capability of storing the present contents of the accumulator while executing the

next instruction. Split-word arithmetic is used in performing arithmetic operations on both halves of a split word at the same time. A split word on the D17B consists of 11 bits. Minimized access timing is the placing of instructions and data in memory so that they are available with minimum delay from the disk memory.

In order to have the D17B computer simulation program simulate the actual computer as closely as possible, all of the foregoing characteristics have been included. As a result of this similarity, the simulation program shows promise for usefulness as a standard for an operational D17B computer. By comparing the results of a test program provided as input to both the hardware and software versions, register and instruction execution malfunctions in the hardware version can be detected.

The D17B computer can be loaded with programs and data from punched tapes. The program and data are punched onto the tape by a tape punch and a tape reader is used to enter this information into the D17B computer. The simulation program is extremely helpful in the preparation of these program tapes which are to be read into the D17B computer. The simulation program has the capability of reading the same punched tapes for input data as are used in loading the D17B computer. The simulation program helps in the preparation of program tapes by detecting and locating invalid symbols punched on the tape and by decoding the program instructions. The simulation program also has the capability to detect addresses (locations in memory) that are out of range of the present program being run. These capabilities have shown the D17B computer simulation program to be very useful.

Thesis Outline. Chapter II of this thesis contains a description of the structure and organization of the simulation program. The functions performed by the main program and subroutines are discussed and a description of the variables used in writing the simulation program is given. Chapter III contains a description of the simulation language which is used as input data for the simulation program. Methods for creating programs to be run on the simulated computer are given and a method for creating a shortened version of the simulation language is presented. Chapter IV contains a listing of the error statements provided by the simulation program. Chapter V contains example programs which have been run on the simulated computer. Several programs are listed which show the types of output that are available from the simulation program. Chapter VI is the concluding chapter and contains recommendations for additions to the simulation program to enlarge its capabilities.

Four appendixes are included with this thesis to provide additional information and clarification to the D17B computer simulation description. Appendix A contains a listing of the simulation program. Appendix B is a compilation of the D17B instruction set and a listing and description of the D17B load codes. Appendix C contains figures for interpreting the simulation program output results for binary, discrete, and voltage outputs. Appendix D supplies information for using the D17B simulation program at AFIT. Also included in Appendix D is a condensed listing of the simulation language.

The description of the D17B computer simulation program presented in this thesis assumes the reader has a basic knowledge of the D17B computer and the procedures for programming it. No attempt is made to describe the D17B computer or to describe D17B programming methods. For information concerning these areas, the reader should refer to references 1 and 4.

References 4, 5, 7, and 8 are the main sources of information used in writing the simulation program. Reference 4 is a training manual for the D17 computer which describes the functions and operations of the computer. Reference 5 is a collection of figures which show pictorially the D17B functions and operations. Reference 7 is an engineering manual with a function breakdown of the logic equations and timing diagrams of the computer operations. Reference 8 is an Air Force Technical Manual containing all the logic equations implemented on the D17B computer.

II. D17B Computer Simulation Program

This chapter describes the organization and structure of the D17B computer simulation program. In writing the simulation program, the plan was to simulate the actual computer as closely as possible. This close correlation between the actual computer and the simulation program makes it possible for a user to use both the computer and simulation program using only one set of programming techniques. However, there are several areas in the simulation program where a quasi-simulation approach was used. The quasi-simulation approach uses the same register inputs and generates the same results, but the methods of obtaining the results differ.

In preparing to write the simulation program, several computer simulation languages were studied, the predominant one being the Computer Design Language (CDL) developed at the University of Maryland. This language consists of computer elements (register, memory, counters, etc.) and is described in the first five chapters of reference 2. Portions of the D17B computer simulation program were written in CDL, but because of the nonavailability of a CDL compiler, a transformation to the FORTRAN language was made.

The simulation program is written in the FORTRAN Extended Language to be run on the Intercom System (teletype) of a CDC 6600 Computer system. Instructions for using the simulation program at AFIT are contained in Appendix D. Appendix A is a listing of the simulation program.

The D17B computer has several codes and addresses which it decodes and uses in loading and executing a program. Karnaugh Maps (Veitch diagrams) of the operation codes, flag store codes, load codes, and channel addresses are shown in Figs. 1 and 2. These codes and addresses appear in the computer in binary form. The operation code is a four bit code used to determine the instruction to be executed. The flag store code is a three bit code which determines where flag store will take place. The load codes are five bit codes used in loading data into the computer. An instruction address is a seven bit code which determines the sector location of the next instruction. The instruction channel address can only be changed by using a transfer (TRA) instruction. A number address is a twelve bit code which consists of a five bit channel designation and a seven bit sector location. Because FORTRAN instructions do not operate on binary data, a correlation between the operation code, flag store code, load code, instruction address, and number address of the D17B computer and a number in the FORTRAN program had to be made. This relationship was made by taking each code or address and changing the binary representation to its equivalent decimal representation. The decimal representation was then used as the designation for the code or address in the FORTRAN program. Included on the diagrams in Figs. 1 and 2 are the binary representation, the quasi-octal representation, and the FORTRAN designation.

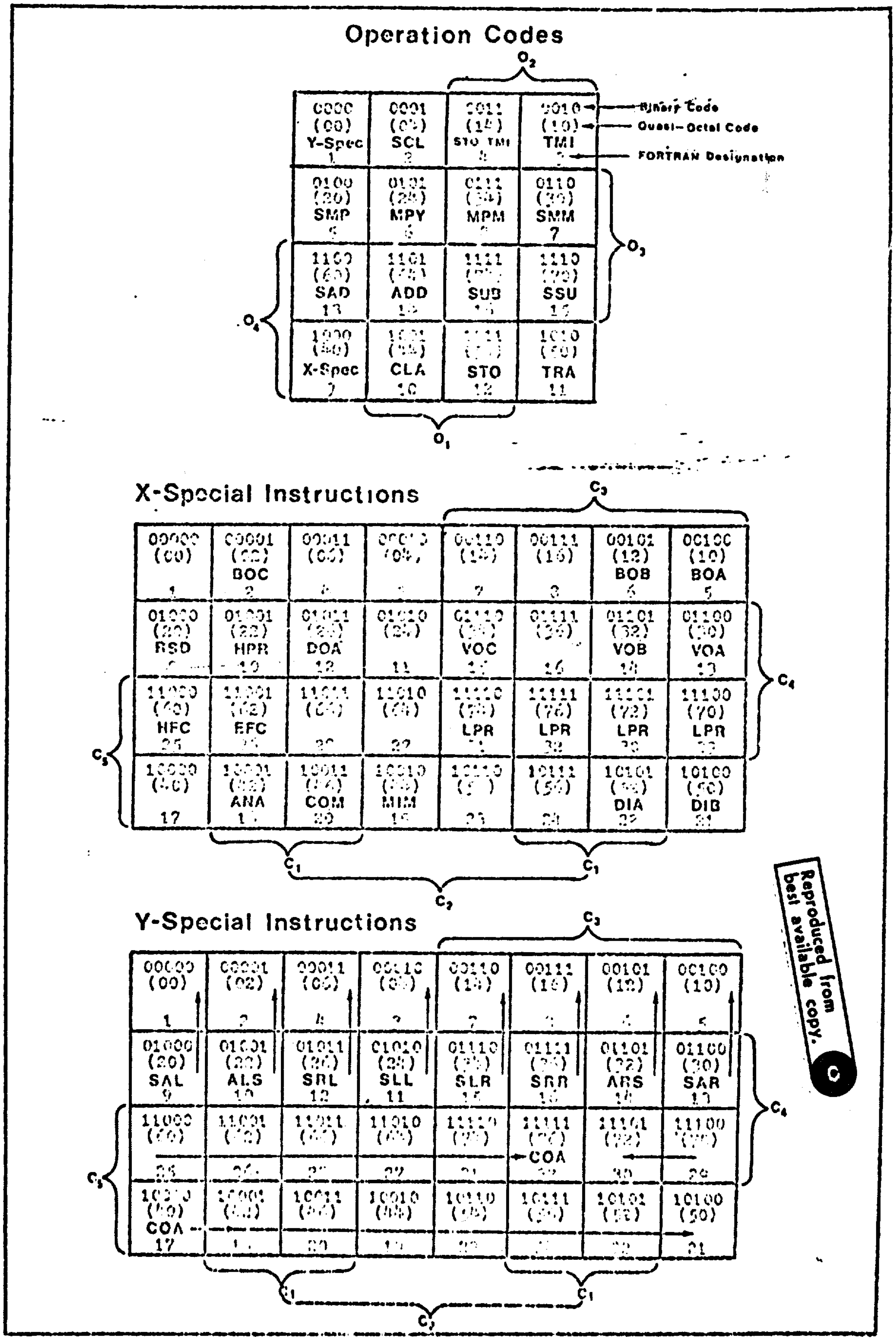


Fig. 1. Veitch Diagrams of Operation Codes and Special Instructions

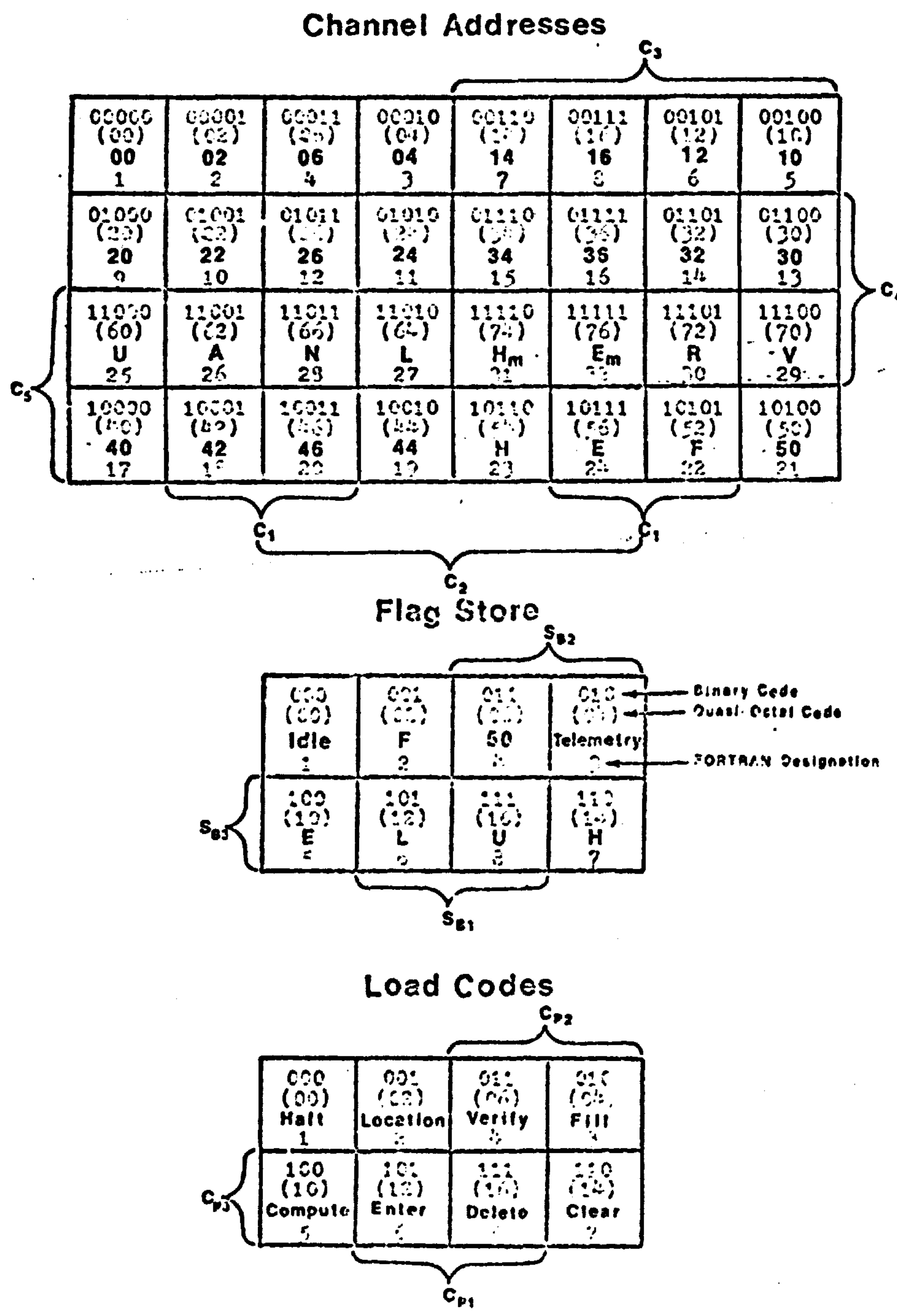


Fig. 2. Veitch Diagrams of Channel Addresses and Flag Store and Load Codes

A quasi-octal representation of the codes and addresses can be made by taking the binary representation and converting to octal. It is necessary to assume pseudo-zero bits in specific locations. The quasi-octal representation is discussed in the D17B Computer Programming Manual (Ref 1:8). The same type of correlation was also used in designating a sector location in the FORTRAN program. Fig. 3 shows the correlation that was made between the F-loop, V-loop, R-loop, E-loop, H-loop and a sector location.

The concept used in writing the simulation program was to have the person using it provide the same data to the program as he would if he were using the actual computer in the laboratory. The switches must be turned to the proper positions to accomplish loading and computing. The data must be error free to successfully execute a program. The type of display (register or memory) is specified by the user.

The D17B computer simulation program consists of a main program and eight subroutines. The main program is a compilation of three distinct sections each of which performs a major function. These three sections are the reading and translation section, the noncompute mode section, and the compute mode section. Fig. 4 shows the program flow between these sections of the main program and the subroutines.

The organization and structure of each of the sections

SECTOR		V-LOOP R-LOOP F-LOOP		E-LOOP		H-LOOP	
MM	FORTRAN	MM	FORTRAN	MM	FORTRAN	MM	FORTRAN
000	1	0	1	0	1	0	1
001	2	1	2	1	2	1	2
002	3	2	3	2	3	2	3
003	4	3	4	3	4	3	4
004	5	0	1	4	5	4	5
005	6	1	2	5	6	5	6
006	7	2	3	6	7	6	7
007	8	3	4	7	8	7	8
010	9	0	1	0	1	10	9
011	10	1	2	1	2	11	10
012	11	2	3	2	3	12	11
013	12	3	4	3	4	13	12
014	13	0	1	4	5	14	13
015	14	1	2	5	6	15	14
016	15	2	3	6	7	16	15
017	16	3	4	7	8	17	16
020	17	0	1	0	1	0	1
021	18	1	2	1	2	1	2
022	19	2	3	2	3	2	3
023	20	3	4	3	4	3	4
024	21	0	1	4	5	4	5
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
170	121	0	1	0	1	10	9
171	122	1	2	1	2	11	10
172	123	2	3	2	3	12	11
173	124	3	4	3	4	13	12
174	125	0	1	4	5	14	13
175	126	1	2	5	6	15	14
176	127	2	3	6	7	16	15
177	128	3	4	7	8	17	16

Fig. 3. Correlation between Minuteman and FORTRAN designation for Sector Track and Multi-word Loops

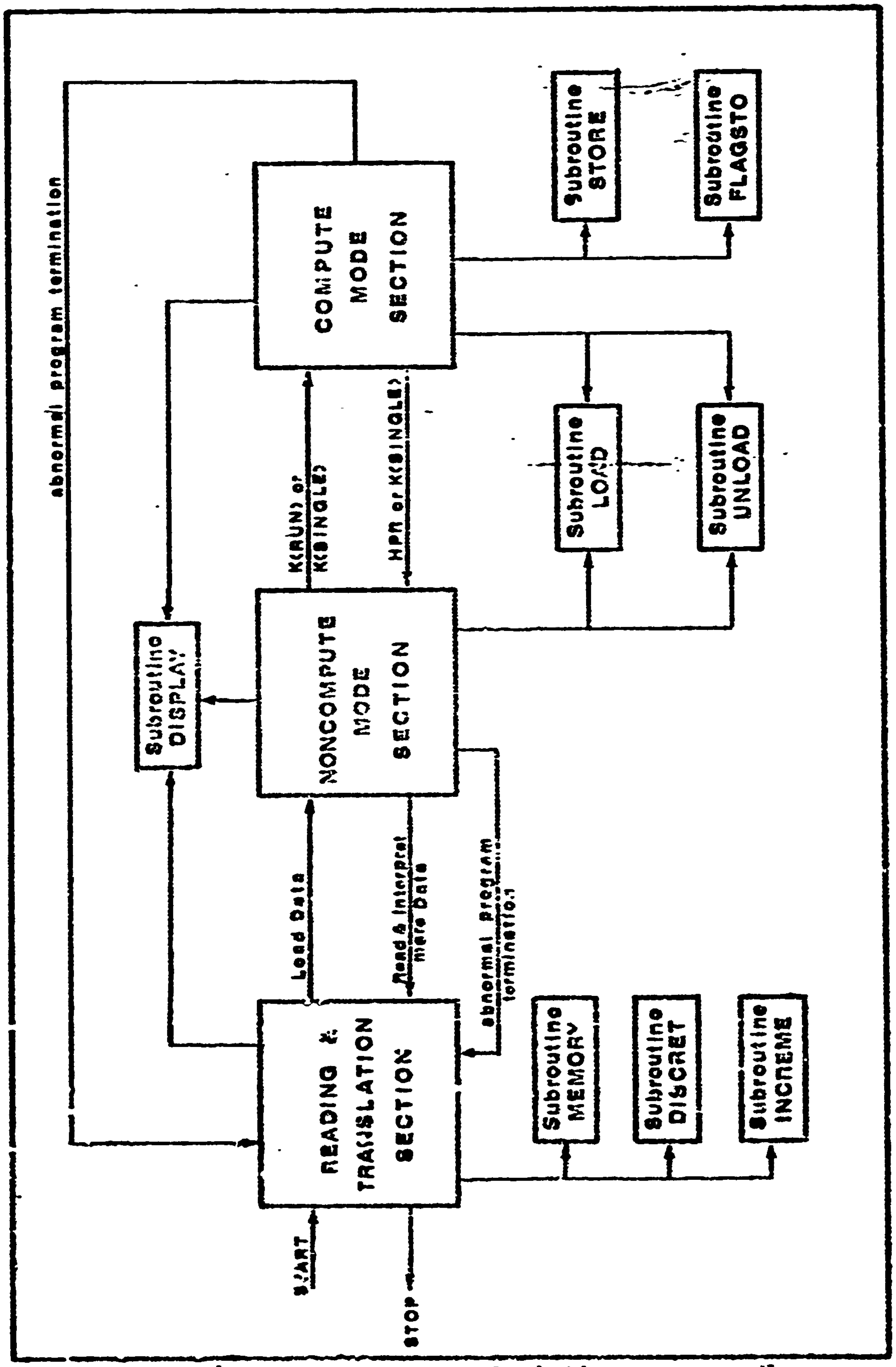


Fig. 4. [illegible] Computer simulation program flow

of the main program will be discussed along with the functions performed by each of the subroutines. The variables used in creating the simulation program are also listed with a short description of how each is used.

Reading and Translation Section. The reading and translation section is the translater and interpreter portion of the simulation program. All input data is read, interpreted, and translated in this portion of the main program. A transfer of operation to the noncompute mode or one of the subroutines is made to utilize this data. The programming language accepted as valid data by the simulation program is described in detail in chapter III of this thesis and will not be discussed in the following description of the reading and translation section.

The reading and translation section is physically located at the beginning of the simulation program. When the simulation program is loaded for execution, execution begins at the start of this section. The first output produced by this section is a heading containing the name of the simulation program, the date, and the time at the beginning of execution. The remainder of the reading and translation section is responsible for the reading, interpreting, and translating of input data. Input data is read as alphabetic or numeric characters (hollerith). This data is then interpreted as octal or binary data, a load code, a switch designation (setting), or a miscellaneous input or command. The miscellaneous inputs or commands are

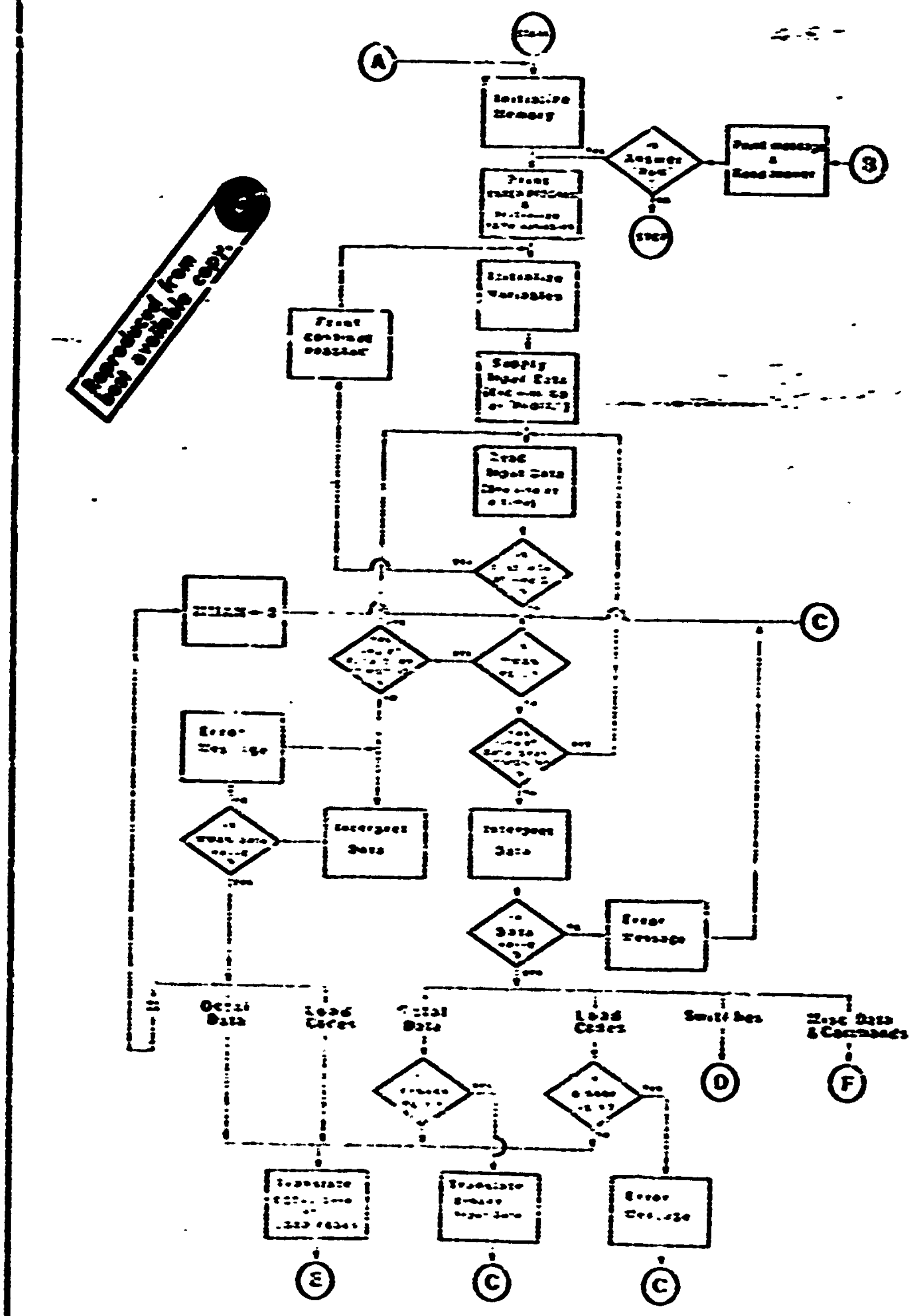


Fig. 5. Reading and Translation Section Flowchart

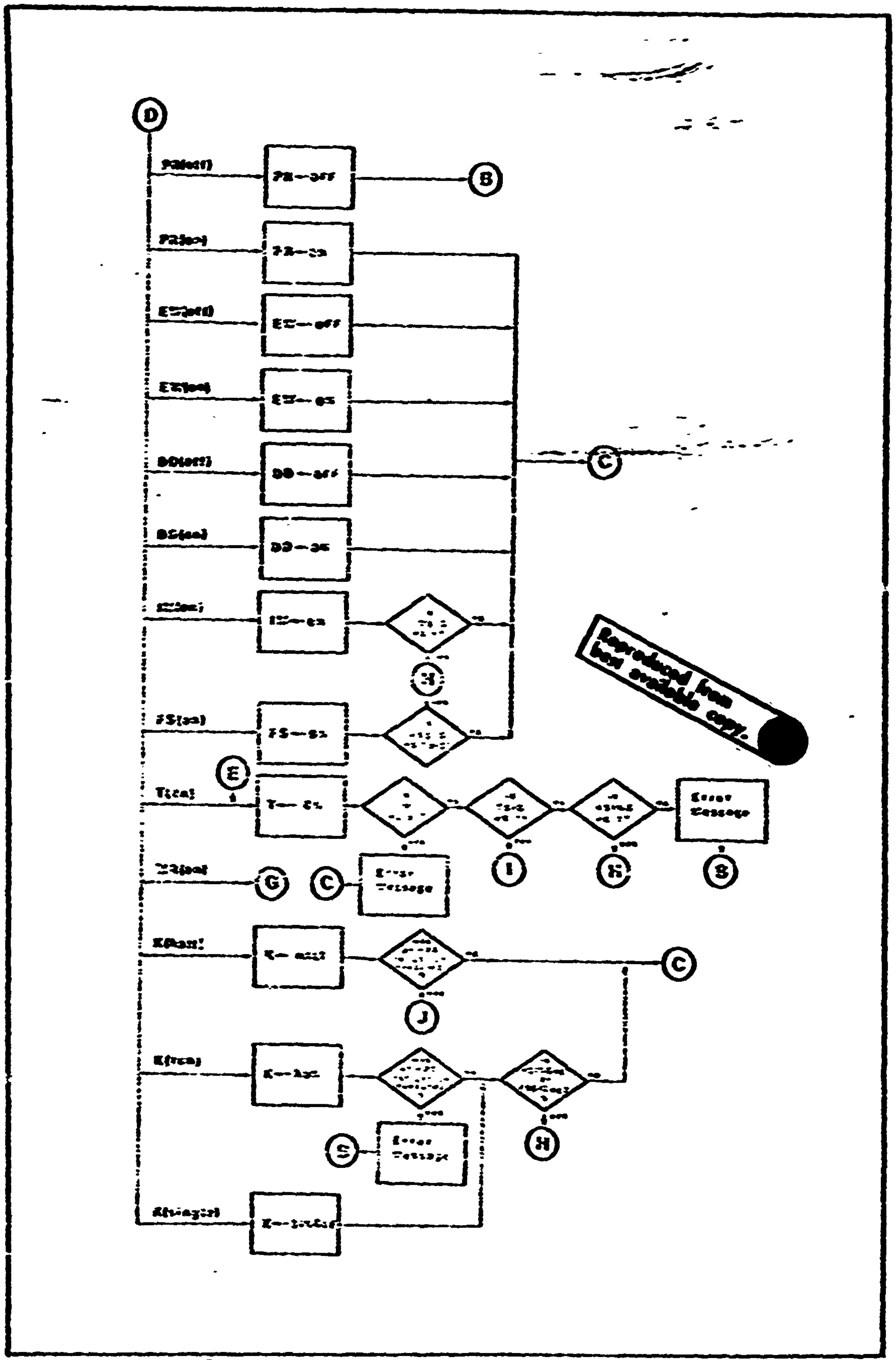


Fig. 6. Switch Interpretation Flowchart

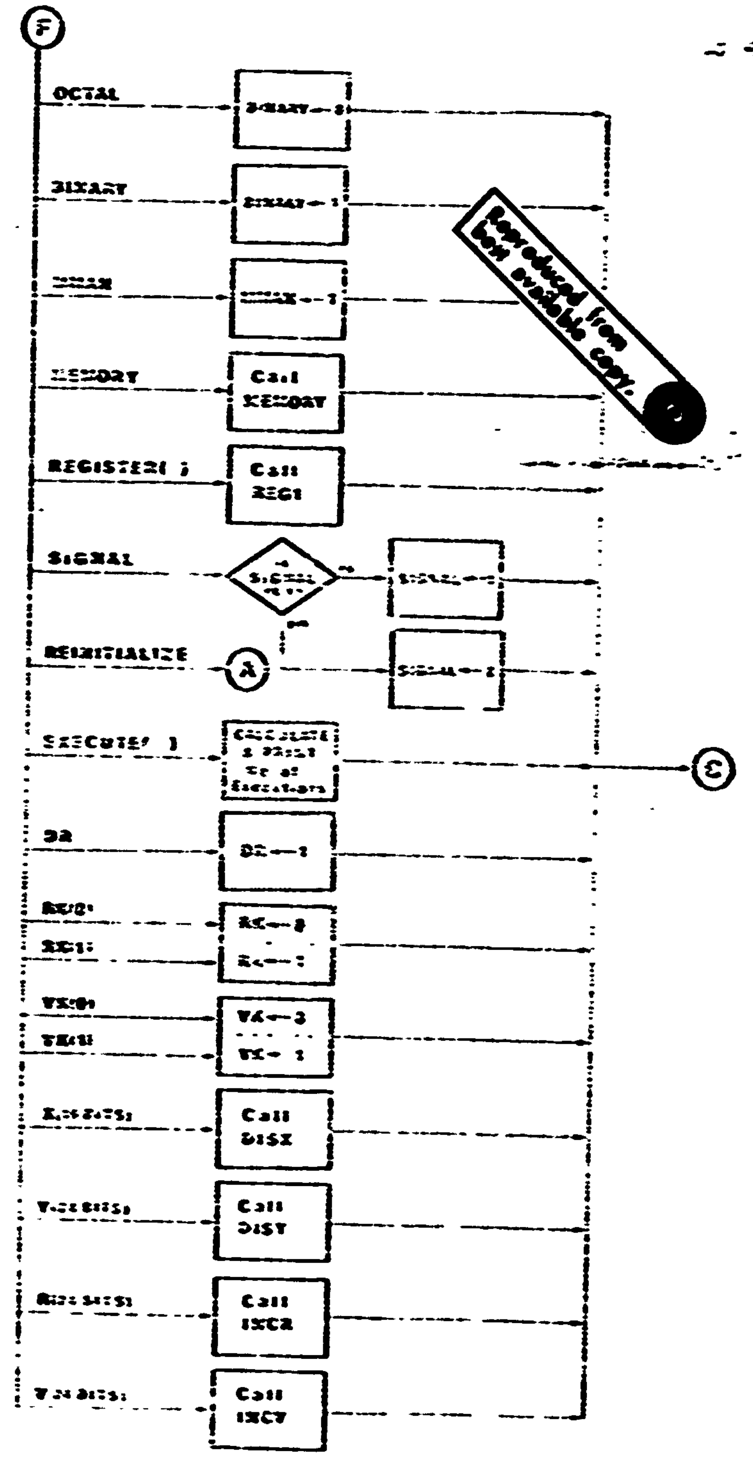


Fig. 7. Miscellaneous Input and Command Flowchart

responsible for a variety of functions which include the following: register and memory display, discrete data, incremental data, setting of flipflops, and mode tracing.

If data is interpreted as octal data, binary data, or a load code, a translation is made from the hollerith representation to binary integer data consisting of 1's and 0's. This binary integer data is then supplied to the noncompute mode of the simulation program where it is utilized. A switch designation results in a switch variable being loaded with the designation. Miscellaneous input or command data results in either the storing of input data or the flagging (setting to 1 or 0) of variables which control program flow.

Flow charts showing the organization and structure of the reading and translation section are shown in Figs. 5 thru 7. Fig. 5 shows the interaction and interpretative capability of the simulation program. Figs. 6 and 7 are extensions of the flowchart shown in Fig. 5 and show the results of interpretation of switch designations and interpretation of miscellaneous inputs and commands.

Non-Compute Mode Section. This section of the simulation program simulates the noncompute mode of the D17B computer. The noncompute mode can be divided into two categories each performing a major function. These two categories are noncompute load and noncompute nonload. The noncompute nonload function comprises the following modes: "prepare to operate", "sync bit counter 1", "sync bit counter 2", "manual halt", and "program halt". In the prepare-to-operate mode, mode control flipflops are initialized. During sync bit counter 1 and 2 modes, a synchronizing between the clock track bit counter and the bits of a word is made. The manual halt mode is used for idling, preparing to load, and preparing to compute. The program halt mode is entered by execution of a program halt instruction. The noncompute load function is made up of the following modes: "wait", "prepare to sample", "sample code", "parity check", and "process code". The wait mode is used for idling while waiting for input data. The prepare to sample mode is entered when data is detected on the input lines. In the sample code mode the input data is read. The parity check mode is used for checking the input data for odd parity. In the process code mode the input data is decoded and processed according to the deciphered code.

Fig. 8 is a flowchart of the noncompute mode which shows the program flow between these modes. This flowchart was used in writing this part of the simulation program. Fig. 8 is drawn as a veitch diagram with the mode control flipflop states as the boolean variables. As an aid in tracing through the noncompute mode section of the simulation program, the mode control flipflops have been included as comment cards. The mode control flipflops are listed as being set to "1" or "0". In the D17B computer these flipflops were actually set, however, they were not needed in the simulation program because of the sequential flow

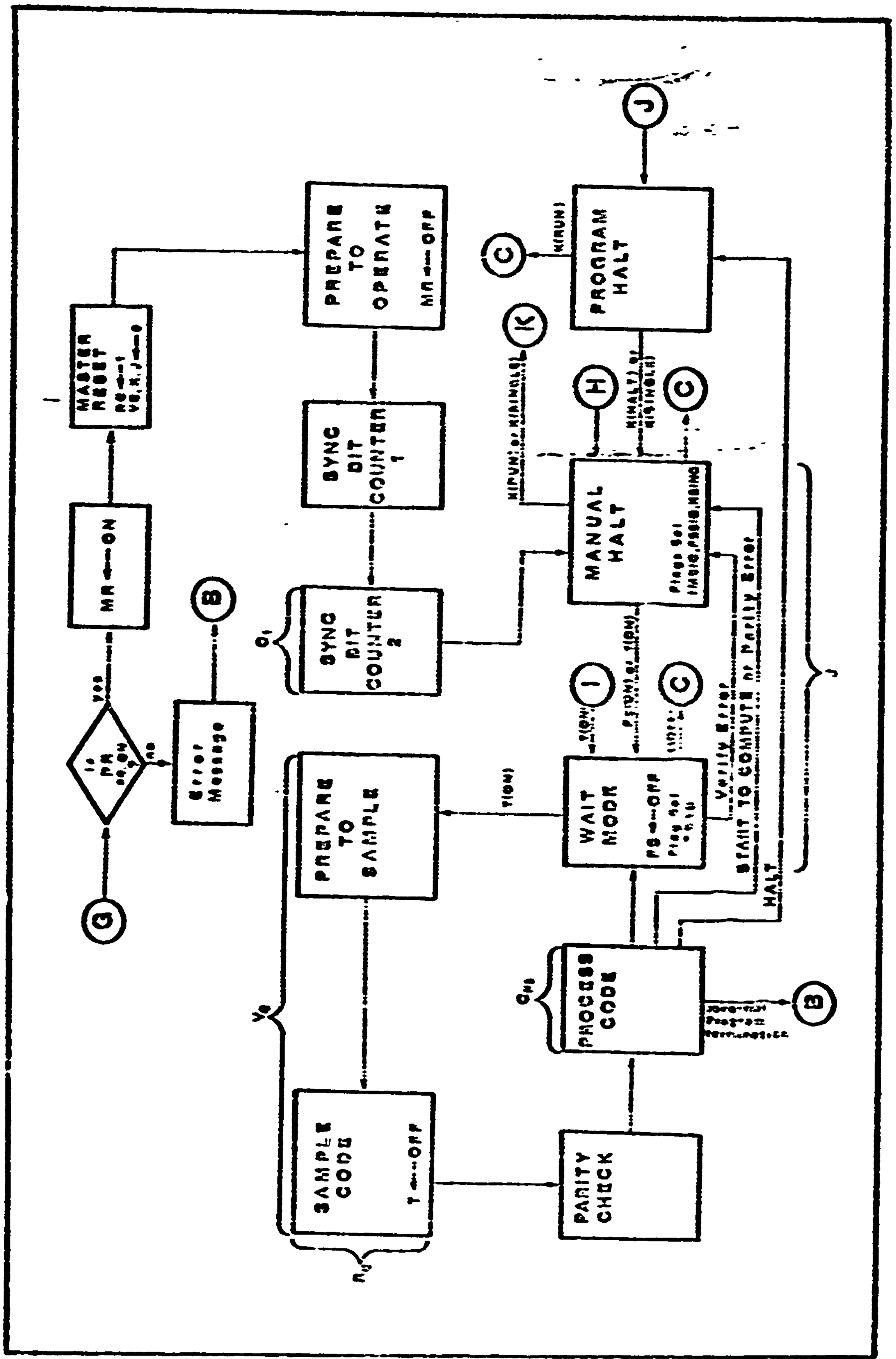


Fig. 8. Non-Compute Mode Section Flowchart

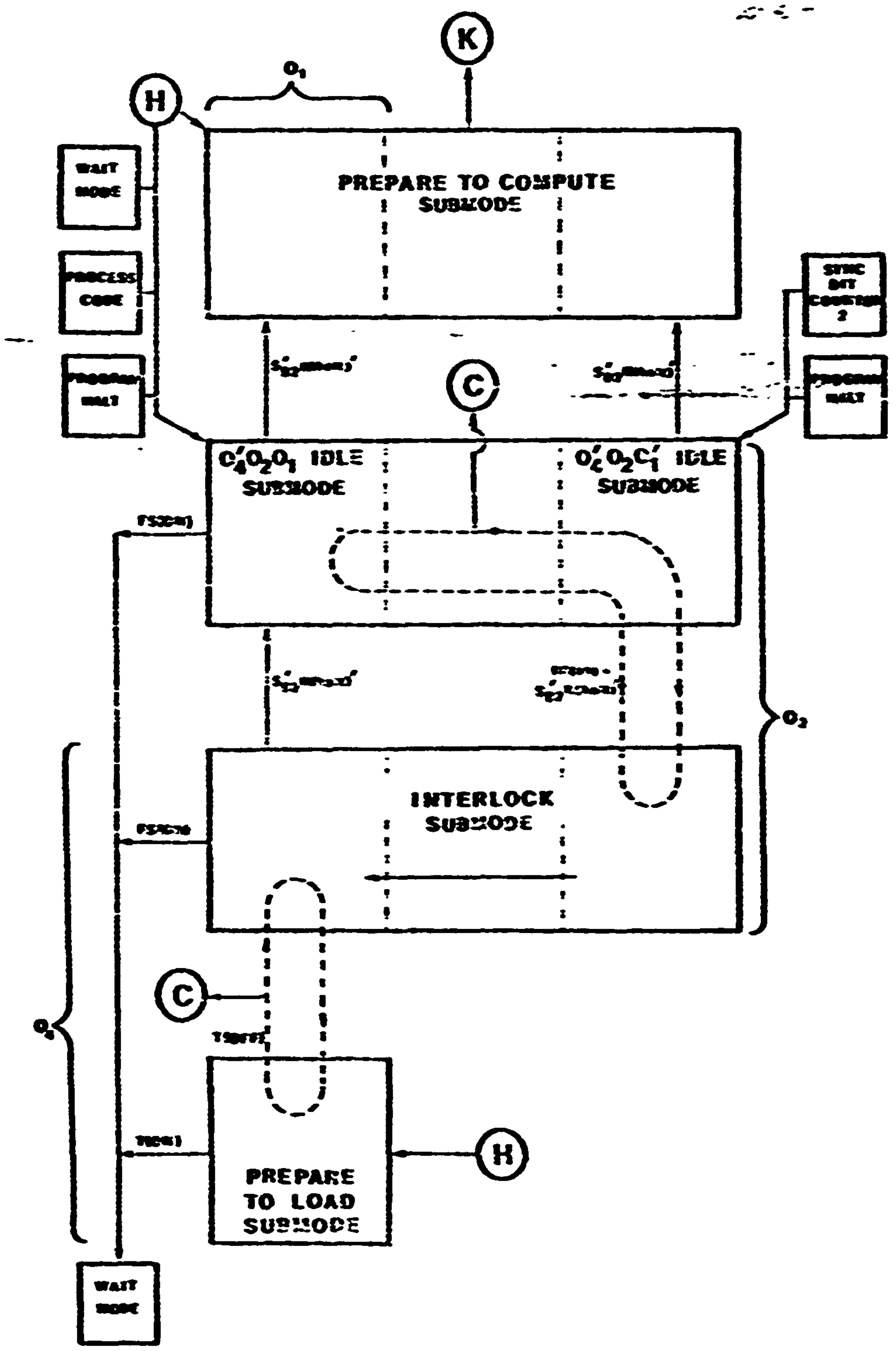


Fig. 9. Manual Halt Mode Flowchart

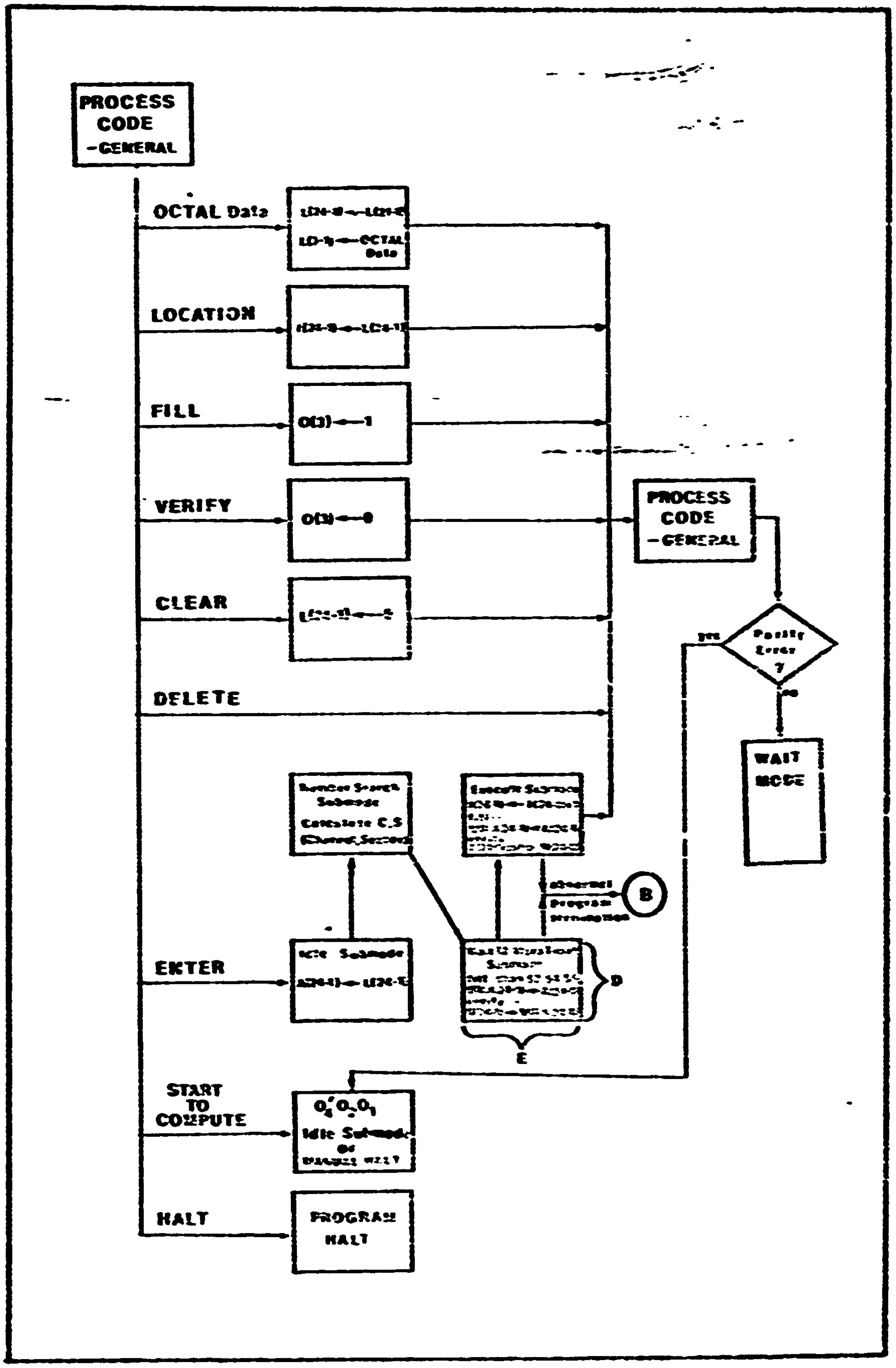


Fig. 10. Process Code Mode Flowchart

that occurs in a FORTRAN program. Two additional figures, Figs. 9 and 10 have been included which give a further breakdown of the program flow in the "manual halt" mode of noncompute nonload and in the "process code" mode of noncompute load.

Compute Mode Section. This section of the simulation program simulates the compute mode of the D17B computer. The compute mode consists of modes which perform five major functions: "number search", "number read", "instruction search", "instruction read", and "execute". The number search, number read, instruction search, and instruction read modes are equivalent to the fetch cycle associated with other computers. The execute mode is equivalent to the execute cycle. Number search and instruction search locate the data word in memory while number read and instruction read unload the located word from memory into a register. Execute results in the execution of one of the 39 instructions in the instruction set of the D17B computer. The compute mode section of the simulation program was written using the above functions. A flowchart which shows the program flow in the compute mode section is given in Fig. 11.

Subroutines. The subroutines associated with the D17B computer simulation program were made for three purposes:

1. Those functions which were needed several times through the program were created as subroutines.

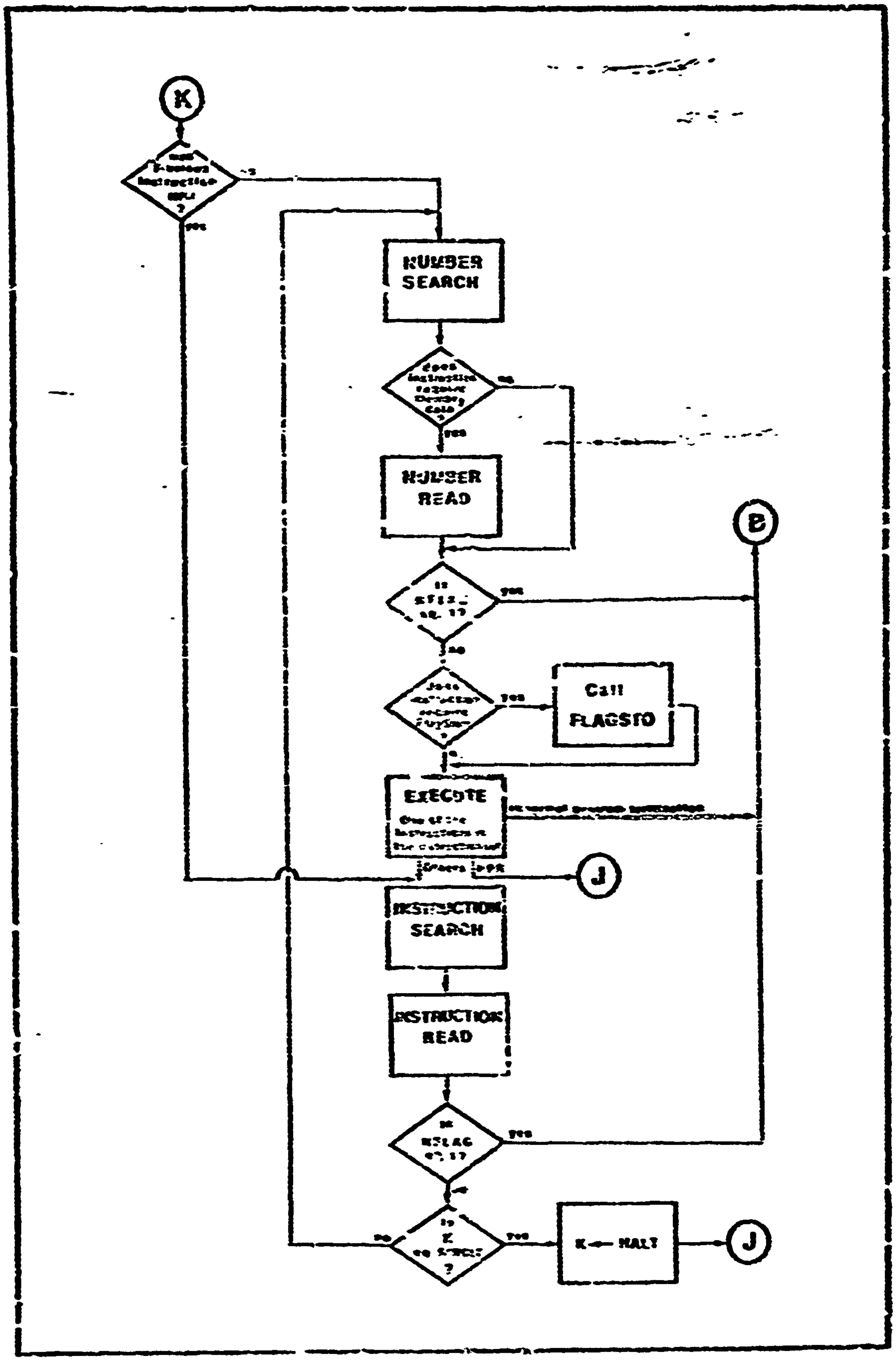


Fig. 11. Compute Mode Section Flowchart

Subroutines falling into this category are Subroutine LOAD, Subroutine UNLOAD, and Subroutine DISPLAY.

2. Those functions which are only called from one place in the main program, but which are of such importance and magnitude that a separate location is beneficial in the organization of the simulation program. Subroutines in this category are Subroutine STORE, Subroutine FLAGSTO, and Subroutine MEMORY.

3. Those functions which will not be used very frequently. Therefore they could be removed from the simulation program if it was determined that they were not really needed. This would result in a decreased memory core size needed for execution of the simulation program. However, to be able to utilize all the instruction set of the D17B computer and all the channel designations these functions had to remain as a part of the simulation program. Subroutines in this category are Subroutine DISCRET and Subroutine INCREM.

A description of the function performed by each of the subroutines will be given. The order of explanation is the order of appearance on the program listing located in Appendix A.

<u>Subroutine STORE</u>. This subroutine performs the store (STO) instruction, which stores the contents of the accumulator in the memory address given in bits 12 thru 1 of the instruction register. A normal loading of memory is performed for all channel designations except the single word channels (A,1,L,U), channel 7[illegible] ([illegible]-loop), and channel

72 (R-loop). The store instruction cannot store in single word loops. Storage in channels 70 and 72 provides the D17B with real time control. Storage in channel 70 results in a whole word addition of the accumulator contents to the addressed word of the [illegible]-loop if the fine countdown flipflop (FC) is "0" set. If FC is "1" set, a normal store takes place. Storage in channel 72 results in a split word addition of the accumulator contents to the addressed word of the R-loop if FC is "0" set. If FC is "1" set no store is allowed. This subroutine can detect erroneous switch settings which terminate the run upon return to the main program.

<u>Subroutine LOAD</u>. This subroutine provides the function of loading the contents of the accumulator into addressed memory locations. The areas of memory that can be loaded by the subroutine are: 20 cold-storage memory channels (channels 00 thru 46), the hot-storage memory channel (channel 50), channel 52 (F-loop), channel 54 (H-loop), channel 56 (E-loop), and channel 60 (U-loop). This subroutine can also detect erroneous switch settings which will terminate the program run. A call is made to this subroutine from both the noncompute mode section and the compute mode section of the main program.

<u>Subroutine UNLOAD</u>. This subroutine performs the function of unloading an addressed word of memory into the N-register. The information unloaded is then used either

as an instruction or an operand number. This subroutine can unload data from all addressable memory channels. Data unloaded from channels 70 and 72 is incremental data used for real time control. If the channel designation is either 70 (V-loop) or 72 (H-loop), one of two possible actions can take place. For the V-loop, if VK (incremental input flipflop) is "0" set then normal unloading occurs, however, if VK is "1" set then the one's complement of the V-loop is unloaded. The same conditions apply to the H-loop and the settings of HK (incremental input ~~flipflop~~). This subroutine can detect out of range conditions for the cold-storage and hot-storage memory channels.

Subroutine [illegible]. This subroutine performs the function of deciphering flag store location bits (bits 17, 18, 19) of the instruction register and storing the contents of the accumulator in the deciphered channel at the sector address associated with the execution of the present instruction. The flag store codes provide for storing in the following channels: hot-storage memory channel (channel 50), channel 52 (F-loop), channel 54 (H-loop), channel 56 (E-loop), channel 60 (M-loop), and channel 64 (L-register). The remaining two flag store possibilities are flag store telemetry signal and flag store idle.

Subroutine DISPLAY. Subroutine DISPLAY provides the simulation program with the capability of displaying the binary contents of all registers and loops. This subroutine has two entry points, ENTRY REG1 and ENTRY REG2. Entry point REG1 is called from the reading and translation section of the main program and performs the function of interpreting the arguments given with the register command. An argument consists of a register designation enclosed in parenthesis immediately following a register command. If a valid argument exists, the variable Registr is set to one. Registr being one set allows the main program to denote a register or loop to be displayed. A call to entry point REG2 is then made to determine if the contents of that register or loop should be displayed. Entry point REG2 checks to see if the register or loop was specified in the register command argument, if not a return is made to the main program. If it was specified then its contents will be displayed as output.

<u>Subroutine MEMORY</u>. This subroutine provides the capability of displaying the contents of memory (channels 00 thru 50) whenever the memory command is specified. Upon entry into this subroutine a check is made of the memory command argument to determine if the display should be in octal or binary. A memory command argument consists of either BINARY or OCTAL being enclosed in parenthesis immediately following a memory command. If no argument was specified, the default condition of OCTAL is used. In displaying the contents of memory, only those portions of memory that have been written into since memory was last initialized will be shown in the output listing. Memory is initialized by writing ten decimal 9's into each word of

memory. This condition is then checked to determine if the contents have changed, and if they have, the contents of that location are printed as output.

Subroutine DISCRET. This subroutine provides the capability of entering discrete data and storing it for use in a program using the discrete input instructions (DIA or DIB). Subroutine DISCRET has two entry points, ENTRY DISX and ENTRY DISY. Entry point DISX is called to interpret and translate X-discrete inputs and store them for use during a program run. 19 bits make up each X-discrete input. A maximum of ten X-discrete input requests is allowed, because the storage area data array in the FORTRAN program is dimensioned for 10.

Entry point DISY is called to interpret and translate Y-discrete inputs and store. Each Y-discrete input consists of 24 bits. A maximum of ten Y-discrete input requests is allowed, because the storage area data array in the FORTRAN program is dimensioned for 10.

Subroutine INCREM. This subroutine provides the capability for entering incremental data into the four words of the V-loop or the four words of the R-loop. Subroutine INCREM has two entry points, ENTRY INCR and ENTRY INCV. Entry point INCR is called to interpret and translate R-incremental inputs and store in R-loop. Each R-incremental input is made up of 24 bits. Four R-incremental input requests fill the R-loop and entering another request causes word 0 of the R-loop to be loaded with the new data. Additional inputs fill word 1, word 2, and word 3 with new data. This sequence can continue indefinitely.

Entry INCV is called to interpret and translate V-incremental inputs and store in V-loop. Each V-incremental input request is composed of 24 bits. Four V-incremental input requests fill the four words of the V-loop and additional input requests cycle through the four words again filling them with new data similar to the R-loop.

Simulation Program Variables. This section contains a compilation and description of the variables used in the D17B computer simulation program. A complete alphabetical listing is made of the variables with descriptions of their main uses within the program. Several of the variables have been used for more than one function. They are described as having no main usage. For these particular variables and several others, the reader should refer to the computer printout and note the use made of them in each instance they have been used. The variables that are in this category are discussed at the end of this section.

Each of the variables in the simulation program are integer variables or have been declared as such in an INTEGER statement, except the variables "Volts" and "Voltage" which are real variables.

The listing and description of the variables is as follows:

A(24) - Accumulator, consists of 24 bits.

AK	- Carry, borrow flipflop.
BINARY	- Binary=0 represents octal designation. Binary=1 represents binary designation.
C(5)	- Operand channel register, consists of 5 flipflops.
CB(5)	- Operand channel buffer register, consists of 5 flipflops, copies the 5 least significant bits of instruction register, for "Y" special instructions.
CHAN	- Contains FORTRAN channel designation.
CODE	- Used mainly for FORTRAN operation code designation.
COMMON(24,2)	- Common registers used for storage and manipulation of information in program.
CP(5)	- Instruction channel register, consists of 5 flipflops.
D(5)	- Discrete output register, consists of 5 flipflops.
DD	- Discrete switch.
DISPLAY	- Set to "P" when output will be disposed to high speed printer; set to "N" otherwise.
DR	- Detector reset flipflop.
E(8)	- E-loop, consists of 8 words.
EW	- Cold-storage memory write switch.
F(4)	- F-loop, consists of 4 words.
FC	- Fine countdown flipflop.
FL	- Initiate loading switch.
FSECT	- Flag store sector designator.
[illegible]	- Set in manual halt mode to [illegible] or [illegible]

	switches to manual halt mode.
G(3)	- Binary output register, consists of 3 flipflops.
H(16)	- H-loop, consists of 16 words.
HALT	- Data word containing hollerith characters "HALT".
HCODE(8)	- Data array which contains hexidecimal code for numbers 8 thru 15.
I(24)	- Instruction register, consists of 24 flipflops.
ICHAN	- Contains calculated FORTRAN designation for instruction channel.
IM	- Mechanical input switch.
IMSIG	- Set in manual halt mode to flag IM switch to manual halt mode.
IREG	- Set to designate memory data to be unloaded into instruction register rather than number register.
ISECT	- Contains calculated FORTRAN designation for instruction sector.
IT(5)	- Input transmission lines, contains the 4 bits of information and a parity bit which make up the input data for octal and load codes.
K	- Compute mode switch, three-position switch (Run, Halt, Single).
KHPR	- Set to 1 by an HPR insruction, used in program control after an HPR instruction has been executed.
KSIG	- Set to 1 by an HPR instruction, used to flag K switch to program halt mode.

KSING	- Set to 1 in manual halt mode to flag K switch or T signal to manual halt mode.
L(24)	- Lower accumulator, consists of 24 bits.
LIST	- Contains number of executions specified, has a default of 50.
LIST1	- Counts the number of executions in compute mode and compares with the number specified.
M(128,21)	- Memory storage array, consists of 2688 words which includes cold-storage and hot-storage memory.
MM(8)	- Data array with ASCII-code designation for load codes.
[illegible]	- Set to 1 when input is to be in ASCII-code; set to 2 when ASCII-code is to be read from Tape2; set to 3 when ASCII-code is to be read from Tape3; reset to 0 at end of ASCII-code.
MR	- Master reset switch, (momentary on type).
N(24)	- Number register, consists of 24 bits.
NBLANK	- Data word containing hollerith character " ".
NCHAN	- Contains calculated FORTRAN designation for operand channel.
NCOL	- Counts input data columns, reset to 0 at count of 73.
NCOMMA	- Data word containing hollerith character ",".
NFLAG	- Set to 1 in a subprogram when fatal error has been encountered, terminates run upon return to main program.

NINES	- Data word containing 9999999999, used to initialize memory and detect out of range conditions.
NLIST	- Data array containing decimal integers 0 thru 9, used in detecting valid argument in execute command.
NLPAREN	- Data word containing hollerith character "(".
NREG(10)	- Data array containing hollerith characters of the registers and loops, used in displaying the contents of loops and registers.
NRPAREN	- Data word containing hollerith character ")".
NSECT	- Contains calculated FORTRAN designation for operand sector.
NUM	- Used in several different applications throughout program.
NW(27)	- Data array used mainly for interpreting data, switch settings, and commands.
NWORD(72)	- An array used to store 72 characters of input data which are to be interpreted.
O(4)	- Operation-code storage register, consists of 4 flipflops.
OFF	- Data word containing hollerith characters "OFF".
ON	- Data word containing hollerith characters "ON".
ONE	- Data word containing octal one.
P(3)	- Phase register, consists of 3 flipflops.
[illegible]	- Used in several different applications

	through program.
PR	- Power on/off switch.
R(4)	- R-loop, incremental input loop consisting of 4 words.
REG(10)	- Array which is set by register command to display those registers and loops given as arguments.
REGIST	- Variable which contains the code of the register or loop that has just changed information.
RI	- Counts number of incremental inputs to R-loop, reset to 1 at count of 5.
RN	- R-loop incremental flipflop, can be set to 1 or 0 by program input.
RUN	- Data word containing hollerith characters "RUN".
SB(3)	- Flag code buffer register, consists of 3 flipflops, used in calculating location to which flagstore will take place.
SECT	- Used mainly to contain FORTRAN sector designation.
SIGNAL	- Variable set by signal command, if set to 1, modes of operation will be traced.
SINGLE	- Data word containing hollerith characters "SINGLE".
T	- Timing signal, (momentary on).
TRAS	- Set to 1 by TRA instruction, used in controlling program operation after a TRA instruction has been executed.
TSIG	- Set to 1 in wait mode, used to flag T signal to wait mode.

U	- U-loop, consists of 1 word.
V(4)	- V-loop, incremental input loop which consists of 4 words.
VI	- Counts number of incremental inputs to V-loop, reset to 1 at count of 5.
VK	- V-loop incremental flipflop, can be set to 1 or 0 by program input.
VO(8)	- Voltage output register, consists of 8 flipflops.
VOLTAGE	- Variable used in calculating analog voltage designated by contents of voltage output register.
VOLTS	- Data array which contains numbers used in calculating voltage output.
WTIME	- Variable which is set to the number of word times required for execution of each instruction.
X(19,10)	- Array which stores X-discrete input data to be used in program run.
XI	- Counts number of X-discrete inputs, maximum of 10 is allowed.
X1	- Counts number of X-discrete inputs used in program, when greater than XI it assumes value of XI.
Y(24,10)	- Array which stores Y-discrete input data to be used in program run.
YI	- Counts number of Y-discrete inputs, maximum of 10 is allowed.
Y1	- Counts number of Y-discrete inputs used in program, when greater than YI it assumes value of YI.

ZERO - Data word containing octal zero.

The major portion of the variables included in this listing have the same name as used in the D17B computer literature. For comparison purposes, the reader is referred to the documents pertaining to the D17B computer listed in the bibliography. (Ref 6:110-114)

The following variables, some of which appear in the above listing, have been used in several different applications in the simulation program: "Code", "Comreg", "Num", "Phase", "Sect", "Voltage", "I1", "I2", "I3", "I4", "I5", and "I6". These variables have been pointed out for those interested in modifying the simulation program or for those interested in implementing the simulation program on a different computer system.

A description of the organization and structure of the D17B computer simulation program has been given in this chapter. The next area to be covered is the simulation language accepted by the simulation program.

III. D17B Computer Simulation Language

This chapter describes the simulation-language understood and accepted by the D17B computer simulation program. The simulation language is the input data to the simulation program. Methods for programming the simulated computed are discussed along with a method for creating a shortened version of the simulation language. Error detection capabilities of the simulation program are presented in chapter IV. Chapter V will present some examples of programs that have been run along with the types of output that are available.

For purposes of presentation, the simulation program language is divided into the following categories: numbers and load codes, switches, and miscellaneous inputs and commands. A description of the elements of the simulation language in each of these categories will be given along with guidelines for using each.

Numbers and Load Codes. The number systems and load codes accepted by the simulation program are:

Octal numbers - 0, 1, 2, 3, 4, 5, 6, 7

Binary numbers - 0, 1

Load Codes - HALT, LOCATION, FILL, VERIFY, COMPUTE, ENTER, CLEAR, DELETE (A description of the Load Codes is given in Appendix B)

Three different representations of the numbers and load codes can be specified. By specifying OCTAL, BINARY, or KHAN, an octal representation, a binary representation, or an ASCII representation of the numbers and load codes can be used. The representation of the numbers and load codes in the three specifications are as follows:

		Octal Representation	Binary Representation	ASCII Representation
Numbers -		0	10000	0
		1	00001	1
		2	00010	2
		3	10011	3
		4	00100	4
		5	10101	5
		6	10110	6
		7	00111	7
Load Codes -		HALT	01000	8
		LOCATION	11001	9
		FILL	11010	Z
		VERIFY	01011	;
		COMPUTE	11100	<
		ENTER	01101	=
		CLEAR	01110	^
		DELETE	11111	?

When OCTAL is specified numbers and load codes must be in the octal representation. When BINARY is specified numbers and load codes must be in the binary representation. When ASCII is specified numbers and load codes must be in the ASCII representation. Program tapes to be run on the D17B computer are in the ASCII representation. The default specification is OCTAL.

To terminate an octal or binary representation, all that is required is to specify another representation. To terminate an ASCII representation requires that the letter "H" be supplied after the last ASCII input symbol. Doing this will cause the program to revert to the octal representation or binary representation which it had before HHAH was specified.

The default specification is assumed if an error results in program termination, if the power switch is turned off, or if REINITIALIZATION is specified.

Switches. With the simulation language in this category it is possible to specify switches and designate a setting or mode. The simulation program accepts these switch designations and provides this information to program variables associated with the switches.

The form for specifying switches is as follows:

Switch(Arg)

where Switch is the designated switch mnemonic name, and Arg is the switch setting or mode position of the switch.

The switches and allowed settings are as follows:

Switch Name	Switch Mnemonic & Settings
Timing Signal	T(ON)
Power On/Off Switch	PR(ON), PR(OFF)
Initiate Loading Switch	PS(ON)

Master Reset Switch	MR(ON)
Cold-Storage Write Switch	EW(ON), EW(OFF)
Discrete Switch	DD(ON), DD(OFF)
Mechanical Input Switch	IM(ON)
Compute Mode Switch	K(HALT), K(SINGLE), K(RUN)

Timing Signal. The timing signal is produced automatically for the octal and ASCII representations. Therefore T(ON) need be used only after each binary representation of a number or load code. The timing signal is turned off by the program.

Power On/Off Switch. The power switch must be turned on after each loading of the binary deck, and this must be done before the master reset switch is turned on to prevent an abnormal program termination. Once the power switch is turned on, it remains on until it is turned off or the program is halted and "END OF PROGRAM" is printed. The default condition for the power switch is OFF.

Initiate Loading Switch. This switch is turned on to initiate loading and puts the simulation program in the wait mode of noncompute. It is a momentary on type switch and is turned off by the program.

Cold-Storage Write Switch. The cold-storage write switch is an on/off type switch that allows writing on the cold-storage channels (channels 00 thru 46) of memory when turned on. When the switch is off, no writing is allowed and any attempt to write will cause an abnormal program termination. Once EW(ON) has been specified this condition will remain until it is turned off or until the program is halted and "END OF PROGRAM" is printed. The default condition for the cold-storage write switch is OFF.

Master Reset Switch. The master reset switch is turned on to initialize certain flipflops, synchronize the bit counter and sector track, load the instruction register with a transfer (TRA) instruction to channel 00, sector 000, and put the simulated computer into the manual halt mode of noncompute. The master reset switch is a momentary on type switch and is turned off by the program.

Discrete Switch. The discrete switch is an on/off type switch that allows writing on the hot-storage channel (channel 50) of memory and allows discrete outputs when turned on. When this switch is off no writing is allowed on channel 50 and any attempt to write will cause an abnormal program termination. Also when this switch is off no discrete outputs can appear which will be reflected in the output listing by the printing of the following statement: "DISCRETE SWITCH IS OFF - DISCRETE OUTPUTS ARE DISABLED". Once DD(ON) has been specified, this condition will remain until it is turned off or until the program is halted and "END OF PROGRAM" is printed. The default condition of the discrete switch is OFF.

Mechanical Input Switch. The mechanical input switch is used for putting the computer in the wait mode of noncompute from the idle submode of manual halt. Whenever IH(ON) is specified it must be followed by an PS(ON) to put the computer in the wait mode. Failure to do this causes an abnormal program termination. The mechanical input switch will be used very infrequently. The mechanical input switch is a momentary on type switch and is turned off by the program.

Compute Mode Switch. The compute mode switch is a three position switch that can be set at RUN, SINGLE, or HALT positions. With the switch set at the HALT position, only functions in the noncompute mode can be performed. Setting the switch at the RUN or SINGLE positions allows the computer to enter the compute mode. If the switch is set to the SINGLE position, one instruction is executed in the compute mode. The next instruction is stored in the instruction register and the simulated computer goes thru the program halt mode to the manual halt mode to wait for further switch settings or conditions. The SINGLE position of the compute mode switch is momentary on type and the program returns the switch to the HALT condition. When the compute mode switch is set to the RUN position, continuous operation occurs in the compute mode until an HPA (halt and proceed) instruction is encountered or the program is abnormally terminated. The compute mode switch has a default condition of HALT.

<u>Miscellaneous Inputs and Commands</u>. The simulation language in this category provides many functions that are unrelated but were not of such importance to warrant being in a category of their own. The functions that will be described in this category are listed as follows:

Register and Memory Display

Incremental Inputs

Discrete Inputs

Mode Tracing

Execution Specifications

Setting of Flipflops

Initialization

<u>Register and Memory Display</u>. The binary contents of any of the registers (A,I,L,M) or loops (U,F,E,H,V,M) can be displayed by use of the register command. The register command has the following form:

REGISTER(Arg)

where Arg is a list of the registers and/or loops to be displayed.

The register command can contain from zero to ten specifications in the argument listing. A valid specification is one of the following letters which when specified will display the contents of the loop or register associated with it whenever the contents of that register or loop change in a program run:

Specification	Register or Loop Displayed
A	Accumulator
I	Instruction Register
L	Lower Accumulator
N	Number Register
U	U-loop (1-word loop)
F	F-loop (4-word loop)
E	E-loop (8-word loop)
H	H-loop (16-word loop)
V	V-loop (4-word ~~input~~ loop)
R	R-loop (4-word input loop)

The arguments of the register command can be separated by commas or blanks or they can be placed one after another.

Examples:	REGISTER(A,I,L,N)	Will display contents of accumulator, instruction register, lower accumulator, and number register.
	REGISTER(VR A)	V-loop, R-loop and accumulator will be displayed.
	REGISTER()	No registers or loops will be displayed.

The default condition for the register command is REGISTER(). The default condition is assumed each time a program run is terminated and more input data is supplied. Therefore if register display is wanted, a register command must be used each time data is entered.

To display the contents of cold-storage and hot-storage channels (channels 00 thru 50) of memory, a memory command is used. The memory command has the following form:

MEMORY(Arg)

where Arg is the type of display requested, either BINARY or OCTAL.

If the argument is not included or incorrectly specified, the default for Arg is OCTAL. The memory command can be used anywhere within the program.

Examples:	MEMORY(BINARY)	Memory display will be in binary.
	MEMORY	Memory display will be in octal.

Incremental Inputs. Because the simulation program does not have real time control capability, provisions were made for entering data in the V-loop and H-loop. This data could then be used in the programming of the simulated computer as though it had been supplied incrementally during real time processing.

The form of the request for entering incremental data is:

Loop(Arg)

where Loop is either V or H and Arg is 24 bits.

For Arg to fill the whole word of the V-loop or H-loop, it should contain 24 or more bits. Any bits above 24 are ignored. Any bits less than 24 results in the least significant bits remaining unchanged. Storage starts with bit 24 and continues towards bit 1 until data is exhausted.

Invalid bits are assumed by the program to be zero and a message is output to this effect. The first incremental input request stores data in word 0 of the V-loop or H-loop (whichever is being filled). The second request in word 1, the third request in word 2, the fourth request in word 3. Additional requests start over with word 0 and repeat the cycle. Each time the program terminates for more data, the program is initialized to start with word 0 again. It is possible by including no data in the argument to skip numbers without changing the previously stored data. For readability blanks are ignored in the argument portion of the request. The data can therefore be arranged in any groupings desired.

Examples: R(000 001 010 011 100 101 110 111) This request fills word 0 of the R-loop with the binary data given in argument.

W() W() W() W(000000111111 000000 111111) This request causes words 0, 1, and 2 of the W-loop to remain unchanged and word 3 to be filled with the binary data given in argument.

<u>Discrete Inputs</u>. Discrete inputs are necessary to supply data for use with the DIA and DIB (Discrete Input A and Discrete Input B) instructions.

The form of the request for entering discrete input is:

Type(Arg)

where Type is the type of discrete input, either X or Y, and Arg is 19 bits for X-discrete inputs and 24 bits for Y-discrete inputs.

For Arg to be valid it must contain the number of bits required for the input type. It can contain more bits than required, because excessive bits above those required are ignored. If not enough bits are supplied, however, the program will use the dat. beyond the request until the proper count is reached.

Invalid bits are assumed by the program to be zero and a message is output to this effect. A maximum of ten X-discrete input requests and ten Y-discrete input requests is allowed before the storage area is filled. Additional requests are ignored. Input requests fill the storage array in sequential order from 1 to 10. Provisions for refilling the discrete storage array once all ten storage areas have been filled is given in the initialization portion of this section.

In using the stored discrete inputs, they are used from 1 to the highest number stored. Additional requests beyond the highest number results in the program using the highest number stored and a message is output to this effect. Whenever an abnormal termination of the program is made or the power switch is turned off, the counter for using discrete inputs is initialized to 1.

Mode Tracing. Mode tracing is used in deciphering the contents of a program. In the noncompute mode, the modes of operation are listed as output. In the compute mode, the instruction being executed is listed as output and a flag store is indicated if it was programmed.

The mode tracing capability is requested by a signal command with the following form:

SIGNAL

Whenever SIGNAL is specified in a program, it flips the representation of a variable from 1 to 0 or 0 to 1 depending upon the value it had when the signal command was given. Node tracing is performed when the signal variable is 1. Whenever a program run is terminated and more input data is supplied, the signal variable is initialized to 0. The signal command used with the register command will give as output a detailed listing of the contents of a program.

Execution Specification. There are numerous occasions when a programmer will inadvertently write a program which loops on itself resulting in execution going on to infinitum. To prevent this from happening in the simulated computer, provisions have been made for counting the number of execution cycles in the compute mode and terminating the program run when the number exceeds a specified amount. The programmer can specify the number of executions allowed by an execute command.

The form of the execute command is as follows:

EXECUTE(Arg)

where Arg is any four digit decimal number from 0000 to 9999.

If no execute command is given, the default value is EXECUTE(0050). Each time the program terminates for more data, the execution cycle counter is initialized to zero. However the number of executions allowed is initialized to the default value only upon an abnormal program termination, power switch turn off, or initialization. The number of executions specified is printed out whenever an execute command is encountered in a program.

<u>Setting of Flipflops</u>. The simulation language described here provides the capability of setting or resetting certain specified control flipflops which can change program flow when encountered. The flipflops that can be set are DR (detector) flipflop, RK (incremental input) flipflop, and VK (incremental input) flipflop.

The status of the DR flipflop is used in the execution of several instructions. It is reset to "0" by program control using the RSD (Reset Detector) instruction. To "1" set the DR flipflop, a command with the following form is used:

DR

The condition of the VK and RK flipflops determine the form of the data unloaded from the V-loop and R-loop respectively. If RK and VK are "0" set, data is unloaded into the M-register in normal form. If VK and RK are "1" set, the one's complement of the data is unloaded into the M-register.

The form for specifying the condition of the VK and RK flipflops is:

Flipflop(Arg)

where Flipflop is either VK or RK, and Arg is 0 or 1.

<u>Initialization</u>. When the binary deck of the simulation program is loaded for execution, memory is initialized by putting ten decimal 9's in every word location, the binary output flipflops are set to a +1 condition on all three lines, the discrete input request counters are set to start counting at 1, and the DR and RK flipflops are "0" set. The

programmer can cause the same initialization to occur by using the initialization command which has the following form:

REINITIALIZATION

Programming Methods. The programming methods presented in this section and the programming examples of chapter V do not discuss methods for programming the D17B computer, but are concerned with methods and examples for programming the simulation program. For a discussion of programming the D17B computer, the reader should refer to the programming manual written for the Minuteman Computer Users Group. (Ref 1)

In the previous section of this chapter, a description of the input language that will be used by the simulation program was given. In this section methods will be described for arranging this language in a program form which can be run on the simulated computer.

The approach for arranging the input language in program form found most advantageous by the author is to visualize a hardware control console with switches for each element of the simulation language. To write a program then requires that the programmer write down the simulation language code for each switch that he would push on the console. This approach works because of the similarity between the simulation program and the hardware version of the computer.

In writing a program to be run on the simulated computer, the programmer is not restricted to any input format. The input is format free and can be entered 72 characters per line. This allows the programmer to write a continuous program with each word of the simulation language separated by a delimiter. A delimiter is a character which fixes the end of a simulation language word. The delimiter required is one blank between each word of the simulation language. Exceptions to the use of a delimiter are that octal data can be grouped and no delimiter is needed for the ASCII representation.

There are three words in the simulation language that must begin in column one. These words are PB(OFF), GO, and $. The two words PB(OFF) and GO signify that the input data is complete and ready to be read and interpreted by the simulation program. Entering PB(OFF) will result in the power switch being turned off at the end of the run. Entering GO causes the simulation program to return for more input data when it is interpreted. A comment line is created by specifying $ in column 1. The program ignores the 71 remaining characters in that line.

Whenever the simulation program is loaded for execution, PB(ON) must be specified before RR(ON). Once the power switch has been turned on it remains on until it is turned off or until the program is halted. EV(ON) and DD(ON) will also remain on until turned off or until the program is halted.

A typical program to be run on the simulated computer will contain switch designations, octal or binary data and load codes, and commands. To enter octal or binary data and load codes, the simulated computer must be in the wait mode of noncompute. One of several ways that this can be accomplished is with the following switch designations:

```
PR(ON) MR(ON) FS(ON)
```

The power switch has now been turned on, the master reset switch has been depressed putting the computer through the "prepare to operate" mode, "sync bit counter 1 & 2" mode where the instruction register is loaded with a transfer (TRA) instruction to channel 00, sector 000, and into the manual halt mode. The initiate loading switch was pushed putting the computer into the wait mode. To enable writing on cold-storage channels of memory and allow the memory to be filled with input data the following would be specified:

```
EW(ON) FILL
```

The cold-storage write switch has been turned on and the fill load code has been specified. The simulated computer is now ready to receive and store data. The following program is a sample D17B computer program to add two octal numbers (14 and 2) and output a telemetry signal of the answer:

```
44010201 ENTER 64020202 ENTER 42402200 ENTER CLEAR 201
LOCATION CLEAR 14 ENTER CLEAR 2 ENTER
```

The addition program and data have been entered into memory. To put the simulated computer into the compute mode requires the following switch designations:

R(RUN) MR(ON)

The compute switch has been set at the RUN position and the master reset switch is depressed again putting the computer through the modes described earlier. Once the manual halt mode is reached, the computer will automatically go to the compute mode and the program will be executed.

The complete program would look as follows:

FR(ON) TR(ON) FS(ON) BS(ON) FILL 44010201 ENTER 64020202 ENTER 42402200 ENTER CLEAR 201 LOCATION CLEAR 14 ENTER CLEAR 2 ENTER R(RUN) MR(ON) .

The above program is complete and upon execution will output the answer (octal 16) via a telemetry signal. The telemetry signal consists of 24 bits (000000 000000 000000 010000).

Entering either FR(OFF) or GO provides a ready signal to the simulation program. This ready signal results in the execution of the input data supplied by the user. If FR(OFF) was specified, then at the end of execution, the teletype would print the following message:

"TO RUN ANOTHER PR [illegible] TYPE 'RUN'; TO STOP TYPE 'HALT' - "

To continue running more programs or more data, the user would type RUN. The system would respond and tell the user to (ENTER PROGRAM). At this point the user would enter more programs or more data.

If GO has been specified then at the completion of the program run the system would respond with the message to (CONTINUE PROGRAM). The user can now execute the same program over again with the same data or new data can be entered. The program written onto the memory will remain there until overwritten or until initialization occurs.

To execute the same program again would require the following:

```
HR(ON) K(RUN)
GO
```

To execute the same program again with new data (20 & 12) would require the following:

```
HR(ON) FS(ON) FILL CLEAR 201 LOCATION CLEAR 20 ENTER
CLEAR 12 ENTER K(RUN) HR(ON)
GO
```

Overwriting the previous program with a program to multiply two numbers (+.04000000 and +.30000000) could be accomplished as follows:

```
HR(ON) FS(ON) FILL CLEAR 1 LOCATION 24020202 ENTER
CLEAR 201 LOCATION 02000000 ENTER 14000000 ENTER
GO
```

This program would produce no results because it was not entered into the compute mode. To execute it would require:

```
MR(ON) M(RUN)
GO
```

Execution of the program produces a telemetry signal of the answer. For a more comprehensive listing of the compute mode portion of the program, the signal and register commands can be used. When using the signal and register commands in the compute mode for the first few times, the number of executions allowed should ~~be lowered.~~ This is done to prevent large amounts of output in case the program has loops. To execute the multiply program using the signal and register commands would require:

```
EXECUTE(0010) SIGNAL REGISTER(A,I,L,R) M(RUN) MR(ON)
GO
```

It is possible to run a program ending with an HPR instruction (such as the previous addition or multiply programs) several times using two different numbers each time. To do this requires specifying M(HALT) when the simulated computer is in the program halt mode and to follow this by FS(ON). Specifying M(HALT) puts the simulated computer in the manual halt mode and specifying FS(ON) puts the simulated computer in the wait mode. If a signal or register command was being used, then SIGNAL REGISTER() should be specified before entering more data. Doing this prevents mode tracing and register display while the new data is being loaded into the computer. The signal and register commands must be respecified before K(RUN) is specified to allow mode tracing and register display in the compute mode. The new data can be entered by specifying a fill load code and following this with the new data. The simulation program can now be put back into the compute mode by specifying K(RUN) followed by MR(ON). This cycle can be repeated as many times as desired. The program for doing this with two sets of data would look as follows:

```
MR(ON) PS(ON) FILL CLEAR 201 LOCATION 20200000 ENTER
04040000 ENTER K(RUN) MR(ON) K(HALT) PS(ON) FILL
CLEAR 201 LOCATION 00300000 ENTER 22340000 ENTER
K(RUN) MR(ON)
GO
```

The previous examples have been entered into the compute mode by specifying MR(ON) K(RUN) or K(RUN) MR(ON). This results in a transfer to channel 00, sector 000 and starting execution at that location. A program can be executed at any starting location by specifying 5000xxxx LOCATION COMPUTE K(RUN). xxxx is the channel address and sector location of the first instruction to be executed. LOCATION puts the transfer instruction 5000xxxx in the instruction register, COMPUTE puts the computer in the manual halt mode and K(RUN) puts the computer in the compute mode for continuous run.

All programs which specify K(RUN) should end with a halt and proceed (HP) instruction. This instruction puts the computer into the program halt mode. If an HPR instruction is not used, the program will continue in the compute mode until an error occurs which will terminate the run or the number of executions exceeds the number specified.

Shortened Version of Simulation Language. The following description of a method for creating shortened versions of the simulation language does not apply to octal data, switches (except compute mode switch), or flipflop settings. These parts of the language have not been included in the discussion.

A shortened version of the input language can be created by the user. To do this requires taking those letters of a simulation language word which are used by the simulation program in interpreting it and using those letters as the input for the word. Additional letters can be added to build a mnemonic form of the word if desired. The following is a listing of those words in the simulation language which can be shortened, a listing of the portions of the word which are used in interpreting it, and an example of a shortened version of the word:

Simulation Language words which can be Shortened	Interpreting Letters	Example of a Shortened Version
HALT	H	HALT
LOCATION	L	LOC
FILL	FI	FILL
VERIFY	V	VER

COMPUTE	CO	COM
ENTER	EN	EN
CLEAR	CL	CL
DELETE	DE	DEL
OCTAL	O	OCT
BINARY	B	BIN
MEAN	MN	MEAN
SINGLE	S	SING
RUN	R	RUN
REGISTER(Arg)	RE(Arg)	REG(Arg)
MEMORY(BINARY)	ME(B)	MEM(B)
MEMORY(OCTAL)	ME(O)	MEM(O)
SIGNAL	S	SIG
EXECUTE(Arg)	EX(Arg)	EXEC(Arg)
REINITIALIZATION	REI	REINIT

The same using conditions apply to the abbreviated version of the language as apply to the full-word version. A blank is needed as a delimiter between each word of the language except octal data. Input data is entered with a free format. All 72 columns can be used for entering input data, however, no language element can be divided between two lines. If a word will not fit on a line, leave the remainder of the line blank and put the word at the first of the next line.

A description of the simulation has been given in this chapter. This language consists of numbers and load codes, switches, and miscellaneous commands. Also discussed was the method for creating programs to be executed by the simulation program. The chapter was concluded with a method for creating a shortened version of the simulation language. The following chapters will give a listing of the erorr detection capability of the simulation program and will present examples of programs that have been run on the simulated computer.

IV. Error Detection

Error detection is one of the outstanding features of the D17B computer simulation program. With this capability the program tapes can be error checked by the simulated computer before they are run on the D17B computer. To successfully load and execute a program on the D17B computer the program has to be error free. At the present time there are no error checks made by the D17B computer except for parity and verifying the contents of memory.

The error detection provided by the simulation program goes beyond checking just program tapes. All input data is checked for validity by comparing the inp[illegible]a symbols against the simulation language symbols. Checks are also made by the simulation program to detect invalid switch settings, addresses that are out of range of the program, and a variety of conditions that are not allowed by the D17B computer.

A listing of the error statements that are provided by the simulation program is given in this section. Included with each statement are possible causes of the error or a further explanation of the error.

Error Statements/Causes of Errors. A listing of the error statements provided by the D17B computer simulation program is as follows:

THE FOLLOWING DATA IS NOT ALLOWED: (Invalid Data)
Input symbols specified are not part of the simulation language.

LOAD CODES MUST BE IN BINARY WHEN BINARY IS SPECIFIED
A different representation for a load code was used when the binary representation was specified.

THE FOLLOWING INPUT DATA IS INVALID: (Invalid Data)
Portions of the input word were interpreted but an improper symbol was encountered disallowing any further interpretation.

COMPUTER IS NOT IN WAIT MODE - DATA CANNOT BE ENTERED - PROGRAM TERMINATED
The program must be in the wait mode of noncompute for data to be entered.

AN FS(ON) SIGNAL MUST FOLLOW AN IK(ON) SIGNAL TO PUT MACHINE IN WAIT MODE - DATA IGNORED
If IK(ON) is specified, the only way to put the computer in the wait mode is to specify FS(ON).

COMPUTE MODE SWITCH SPECIFIED INCORRECTLY
Only RUN, SINGLE, or HALT can be used as modes for the compute switch.

THE FOLLOWING INPUT DATA ON HHAN TAPE IS INVALID - (Invalid Symbol)
When HHAN has been specified, input data must be in the ASCII representation.

EXECUTE ARGUMENT SPECIFIED INCORRECTLY - DEFAULT VALUE OF 50 ASSUMED
The argument of the execute command must contain four decimal digits to be valid.

COLD-STORAGE WRITE SWITCH SPECIFIED INCORRECTLY
Only ON or OFF can be used as the settings for the cold-storage memory write switch.

DISCRETE SWITCH SPECIFIED INCORRECTLY
Only ON or OFF can be used as the settings for the discrete switch.

POWER ON/OFF DATA IS INCORRECT
Only ON or OFF can be used as the settings for the power on/off switch.

CHANNEL SPECIFIED CANNOT BE LOADED - PROGRAM TERMINATED
Only channels 00 thru 56 can be loaded by an enter load code.

MR(ON) CANNOT BE SPECIFIED WITH PR(OFF) - PROGRAM TERMINATED
PR(ON) must be specified before MR(ON).

M(HALT MUST BE SPECIFIED BEFORE M(RUN) AFTER AN HPR INSTRUCTION - PROGRAM TERMINATED
An HPR instruction puts the computer into the program halt mode and M(HALT) must be specified to ~~get out~~ of the program halt mode and into the manual halt mode.

A TRANSFER IS NOT ALLOWED TO L-REG, V-, OR R-LOOPS - PROGRAM TERMINATED
Channels 64, 70, and 72 cannot be used with a TRA (Transfer) instruction).

THE X-INSTRUCTION REQUESTED IS NOT AN INSTRUCTION
Not all possibilities for X-special instructions have been wired into the computer, you have specified one of these areas for execution. Check the Operation Code veitch diagram.

STORAGE CANNOT TAKE PLACE IN COLD-STORAGE CHANNELS IF COLD STORAGE WRITE SWITCH IS OFF - PROGRAM TERMINATED
EW(ON) must be specified before writing can take place on cold-storage memory channels.

STORAGE IS NOT ALLOWED IN SINGLE-LOOPS (A,I,L, OR U) - PROGRAM TERMINATED
The STO (Store) instruction cannot store in single-word loops.

STORAGE CANNOT TAKE PLACE IN CHANNEL 50 IF DISCRETE SWITCH IS OFF - PROGRAM TERMINATED
DD(ON) must be specified before writing can take place on

channel 50 (hot-storage memory channel).

COLD-STORAGE MEMORY CANNOT BE LOADED IF COLD-STORAGE WRITE SWITCH IS OFF - PROGRAM TERMINATED
EW(ON) must be specified before writing can take place on cold-storage memory channels.

HOT-STORAGE MEMORY CANNOT BE LOADED IF DISCRETE SWITCH IS OFF - PROGRAM TERMINATED
DD(ON) must be specified before writing can take place on channel 50 (hot-storage memory channel).

FLAGSTORE IS ALLOWED ONLY IN L-REG WHEN STORE INSTRUCTION IS TO CHAN 50, F, H, OR E-LOOPS - PROGRAM TERMINATED
If a STO (Store) instruction is to channels 50, 52, 54, or 56, a flag store is allowed only to channel 64.

OPERAND ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED
An area of memory has been designated for an operand, but no data has been loaded there.

INSTRUCTION ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED
A memory word has been designated as the next instruction, however no data has been loaded or stored into that memory word.

REGISTER DISPLAY REQUEST IS INVALID - (Invalid Register Display Request)
Register display command is missing a left parenthesis.

(Invalid Symbol) IS NOT A VALID REGISTER DISPLAY ARGUMENT
A symbol other than one of the ten register display symbols was used.

V. Programming Examples

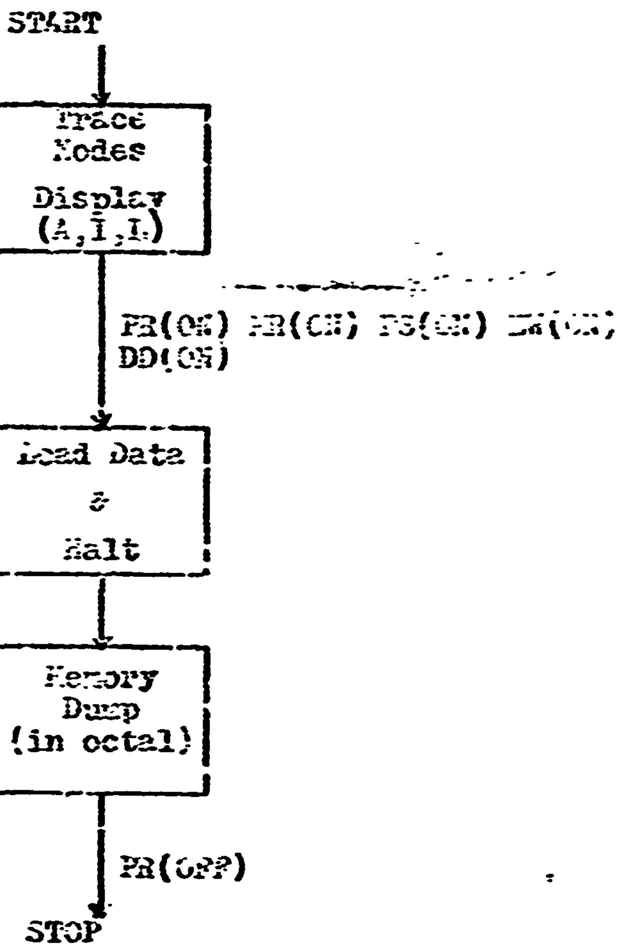
This chapter is concerned with programs ~~that~~ have been run on the D17B computer simulation program. Seven different example programs and flow charts will be presented which show the types of output that can be realized. In the course of presenting these seven examples, the majority of the instructions in the instruction set of the D17B computer will be used.

Example Program Number 1. In this example, ~~the~~ type of output available in the noncompute mode is shown. This example shows the way load codes are used and the way in which data is loaded into memory.

Signal tracing and register display in the noncompute mode will be used very infrequently because of the large amounts of output even for a small program. However, for someone learning the operation of the D17B computer the output displayed by these two commands can be used as a teaching aid.

A memory display request is made at the end of this example. The memory dump that appears on the listing has a channel and sector designation on the left. This address is the memory address of the octal word appearing in the first column of that row. The second column in the same row contains the memory word of the next sector location. The same applies to all remaining words in that row. If a sector location in a row has not been loaded with data it appears on the listing as a word containing all 7's. The rows in which none of the sector locations are loaded with data do not appear in the memory dump listing.

The flowchart for example program no. 1 is as follows:


START
Trace Modes Display (A,I,L)
PR(ON) HR(ON) PS(ON) [illegible](ON)
DD(ON)
Load Data & Halt
Memory Dump (in octal)
PR(OFF)
STOP

```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
 WISE TYPE "N" - N

        ***************************************************************
        **                                                           **
        **                     D17B COMPUTER                         **
        **                  SIMULATION PROGRAM                       **
        **                                                           **
        **     DATE=  01/24/72                  TIME=  19.43.00      **
        **                                                           **
        ***************************************************************

                  **  PRINTOUT OF INPUT PROGRAM  **
 (ENTER PROGRAM)

 SIGNAL REGISTER(A,I,L) PR(ON) MR(ON) PS(ON) EV(ON) ED(ON) FILL
 CLEAR 01234567 ENTER HALT MEMORY(OCTAL)
 PR(OFF)


                    **  RESULTS OF SIMULATION  **

 SIGNAL ON - MODES WILL BE TRACED

 POWER HAS BEEN TURNED ON

 MASTER RESET SEQUENCE
 PREPARE TO OPERATE MODE
 SYNC BIT COUNTER 1 MODE
 SYNC BIT COUNTER 2 MODE
 I(24-1) = 101 000 000 000 000 000 000 000

 C(4) T(2)S(1) IDLE SUB-MODE OF MANUAL HALT
 INTERLOCK SUB-MODE OF MANUAL HALT
 PREPARE TO LOAD SUB-MODE OF MANUAL HALT
 CIRCULATES BETWEEN PREPARE TO LOAD AND INTERLOCK SUB-MODES OF MANUAL
  HALT
 WAIT MODE

 PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - FILL
 WAIT MODE

 PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - CLEAR
 L(24-1) = 000 000 000 000 000 000 000 000
 WAIT MODE

 PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - OCTAL
 L(24-1) = [illegible] 000 000 [illegible] 000 000 000
 WAIT MODE
```

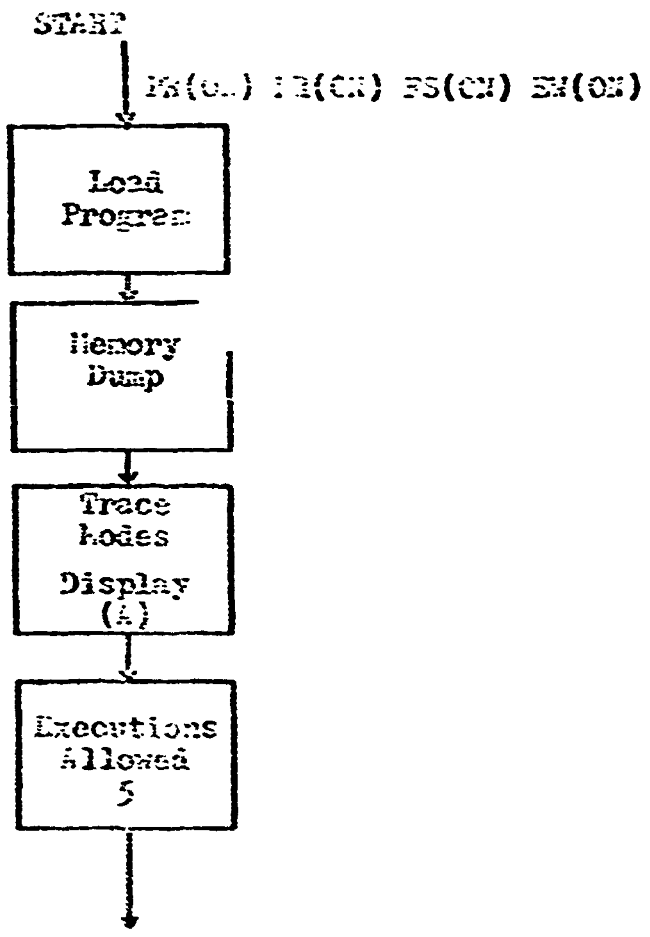
```
PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 000 000 000 000 000 000 001
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 000 000 000 000 000 001 010
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 000 000 000 000 001 010 011
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 000 000 000 001 010 011 100
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 000 000 001 010 011 100 101
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 000 001 010 011 100 101 110
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - OCTAL
L(24-I) = 000 001 010 011 100 101 110 111
WAIT MODE

PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - ENTER
ENTER - IDLE SUB-MODE OF FILL-VERIFY
L(24-I) = 000 001 010 011 100 101 110 111
ENTER - [illegible] SUB-MODE OF FILL-VERIFY
ENTER - [illegible] OF FILL-VERIFY
ENTER - [illegible]
L(24-I) = 1[illegible]1 000 [illegible] 001
WAIT MODE
```

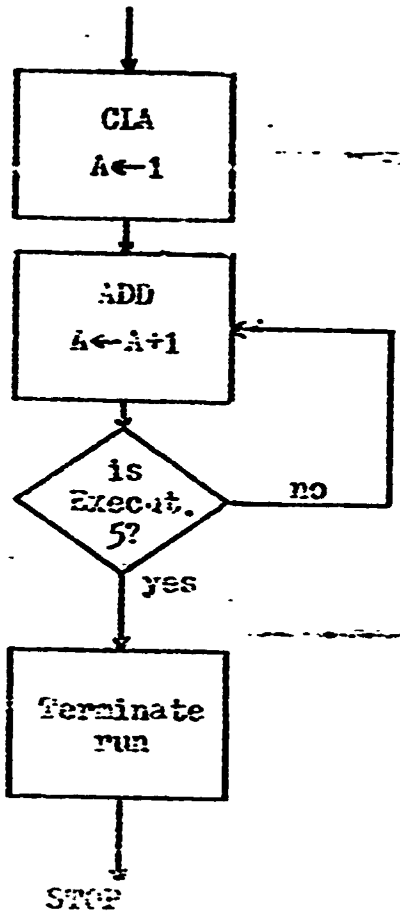
```
PREPARE TO SAMPLE MODE
SAMPLE CODE MODE
PARITY CHECK MODE
PROCESS CODE - HALT
PROGRAM HALT MODE

S(4)"S(2)S(1)" IDLE SUB-MODE OF MANUAL HALT
INTERLOCK SUB-MODE OF MANUAL HALT
PREPARE TO LOAD SUB-MODE OF MANUAL HALT
CIRCULATES BETWEEN PREPARE TO LOAD AND INTERLOCK SUB-MODES OF MANUAL
 HALT


** MEMORY DUMP **

CHAN  SECT
 00   000       01234567       77777777       77777777       77777777

** END OF MEMORY DUMP **      (PORTIONS OF MEMORY NOT LISTED CONTAIN NO
 INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN)


POWER HAS BEEN TURNED OFF
TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT


     **  END OF PROGRAM.        EXECUTION TIME:     .859  SEC
15.51.37.STOP
```

Example Program Number 2. This example consists of an addition ripple program that has been written using the three different representations for input data. Each program has been loaded and executed separately. The addition ripple program loads 1 into the accumulator and keeps adding 1 to the contents of the accumulator. Because this program loops on itself, the execute command has been used to stop the executions at a count of five.

—The initialization and memory commands are used with each representation. Mode tracing ~~and register~~ display are used in the compute mode portion only.

The flowchart for example program no. 2 is as follows:




CLA
A←1
ADD
A←A+1
is Execut. 5?
no
yes
Terminate run
STOP

```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
WISE TYPE "N" - N

         ******************************************************************
         **                                                              **
         **                       5173 COMPUTER                          **
         **                    SIMULATION PROGRAM                        **
         **                                                              **
         **      DATE=  01/24/72                   TIME=  19.54.45       **
         **                                                              **
         ******************************************************************

                      ** PRINTOUT OF INPUT PROGRAM **
(ENTER PROGRAM)

5 ADDITION RIPPLE PROGRAM IN OCTAL REPRESENTATION

PR(ON) SP(ON) PS(ON) SI(ON) FILL.
44010002 ENTER 54010002 ENTER CLEAR 1 ENTER
MEMORY SIGNAL REGISTER(A) EXECUTE(0005) E(RUN) MR(ON)
PR(OFF)


                       ** RESULTS OF SIMULATION **


** MEMORY DUMP **

CHAN  SECT
 00   000        44010002        54010002        00000001        77777777

** END OF MEMORY DUMP **      (PORTIONS OF MEMORY NOT LISTED CONTAIN NO
 INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN)


SIGNAL ON - MODES WILL BE TRACED

NO. OF EXECUTIONS SPECIFIED =     5
MASTER RESET SEQUENCE
PREPARE TO OPERATE MODE
SYNC BIT COUNTER 1 MODE
SYNC BIT COUNTER 2 MODE

S(4)'S(3)S(1)' IDLE SUB-MODE OF MANUAL HALT
PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT


COMPUTE MODE

TRANSFER INSTRUCTION - (TRA)

CLEAR & ADD INSTRUCTION - (CLA)
A(24-1) = 000 000 000 000 000 000 000 001

ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 010
```

```
ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 011

NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
 TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - RUN

                  ** PRINTOUT OF INPUT PROGRAM **
(ENTER PROGRAM)

REINITIALIZE
GO


                  ** RESULTS OF SIMULATION **

MEMORY HAS BEEN INITIALIZED

                  ** PRINTOUT OF INPUT PROGRAM **
(ENTER PROGRAM)

5 ADDITION RIPPLE PROGRAM IN BINARY REPRESENTATION

PR(ON) ME(ON) FS(ON) EN(ON) BINARY
11010 T(ON) 00100 T(ON) 00100 T(ON) 10000 T(ON) 00001 T(ON) 10000 T(ON)
10000 T(ON) 10000 T(ON) 00010 T(ON) 01101 T(ON) 10110 T(ON) 00100 T(ON)
10000 T(ON) 00001 T(ON) 10000 T(ON) 10000 T(ON) 10000 T(ON) 00010 T(ON)
01101 T(ON) 01110 T(ON) 00001 T(ON) 01101 T(ON)
MEMORY SIGNAL REGISTER(A) EXECUTE(0005) M(ON) MS(ON)
PR(OFF)


                  ** RESULTS OF SIMULATION **


** MEMORY DUMP **

CHAN  SECT
 00   000       44010002       64010002       00000001       77777777

** END OF MEMORY DUMP **      PORTIONS OF MEMORY NOT LISTED CONTAIN NO
 INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN)


SIGNAL ON - MODES WILL BE TRACED

NO. OF EXECUTIONS SPECIFIED =    5
MASTER RESET SEQUENCE
PREPARE TO OPERATE MODE
SYNC BIT COUNTER 1 MODE
SYNC BIT COUNTER 2 MODE

S(4)'S(2)'S(1)' IDLE SUB-MODE OF NORMAL HALT
PREPARE TO COMPUTE SUB-MODE OF NORMAL HALT


COMPUTE MODE
```

```
TRANSFER INSTRUCTION - (TRA)

CLEAR & ADD INSTRUCTION - (CLA)
A(24-1) = 000 000 000 000 000 000 000 001

ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 010

ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 011

NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
 TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - RUN
                    ** PRINTOUT OF INPUT PROGRAM **
(ENTER PROGRAM)

REINITIALIZE
GO


                    ** RESULTS OF SIMULATION **

MEMORY HAS BEEN INITIALIZED

                    ** PRINTOUT OF INPUT PROGRAM **
(ENTER PROGRAM)

5 ADDITION RIPPLE PROGRAM IN ASCII REPRESENTATION

PR(ON) ME(ON) PD(ON) EV(ON) [illegible]
Z 44010002 : 64010002 : ^ 1 : H
MEMORY SIGNAL REGISTER(S) EXECUTE(0005) E(RUN) RS(ON)
PR(OFF)


                    ** RESULTS OF SIMULATION **


INPUT SOURCE - [illegible] TAPE


** MEMORY DUMP **

CHAN  SECT
 00   003       44010002       64010002       00000001       77777777

** END OF MEMORY DUMP **      (PORTIONS OF MEMORY NOT LISTED CONTAIN NO
 INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN)
```

```
SIGNAL GO - MODES WILL BE TRACED

NO. OF EXECUTIONS SPECIFIED =    5
MASTER RESET SEQUENCE
PREPARE TO OPERATE MODE
SYNC BIT COUNTER 1 MODE
SYNC BIT COUNTER 2 MODE

B(4)'B(2)B(1)' IDLE SUB-MODE OF MANUAL HALT
PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT


COMPUTE MODE

TRANSFER INSTRUCTION - (TRA)

CLEAR & ADD INSTRUCTION - (CLA)
A(24-1) = 000 000 000 000 000 000 000 001

ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 010

ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 011

NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
 TO RUN ANOTHER PROGRAM TYPE 'GO' ; TO STOP TYPE 'HALT' - HALT


    ** END OF PROGRAM          EXECUTION TIME=    1.125  SEC
20.09.19.STOP
```

Example Program Number 3. This example is an arithmetic program that uses a COA (character output) subroutine to output the answer as eight octal digits. The COA subroutine was developed by the Systems Laboratory Group at Tulane University. (Ref 1:27,28)

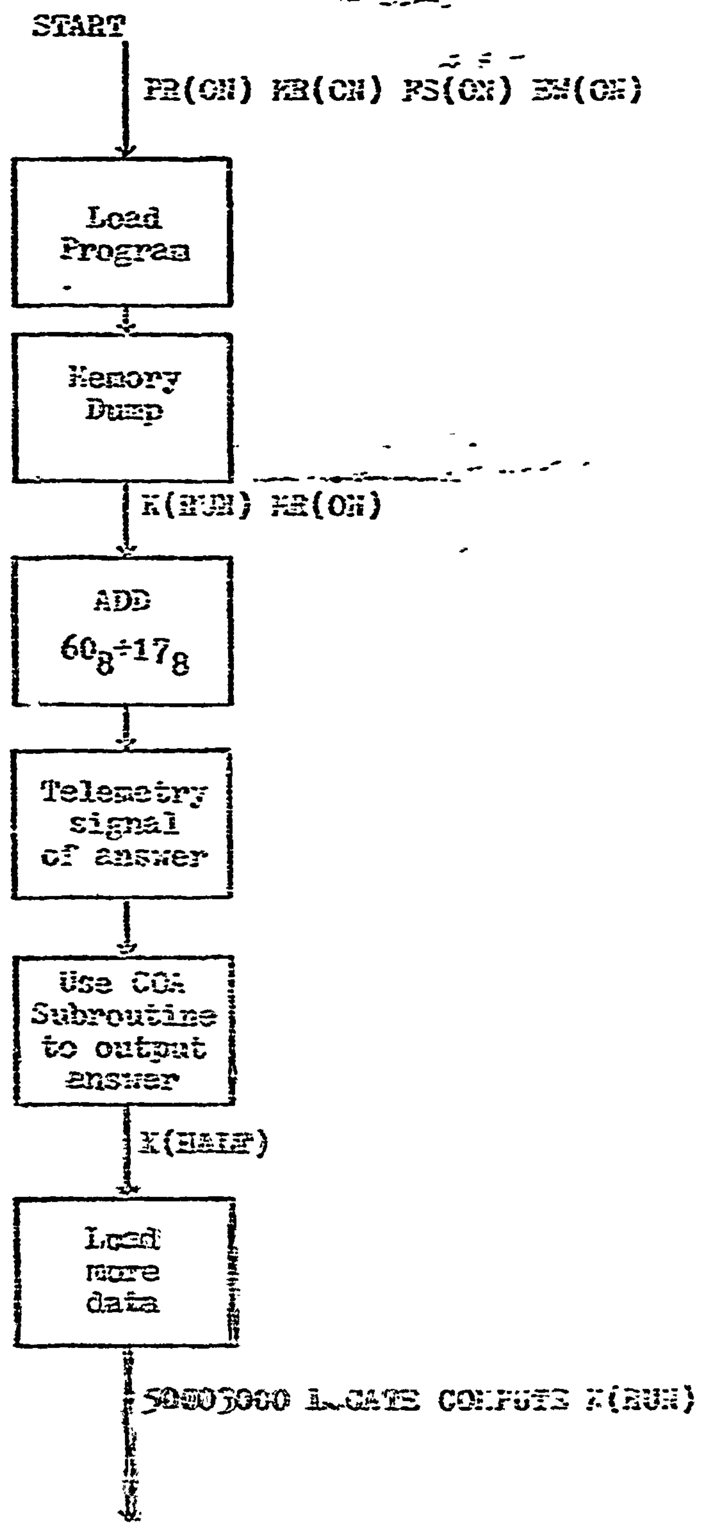
The arithmetic program consists of seven arithmetic routines with starting locations at the program address listed below. Each routine needs two data numbers starting at the data addresses given.

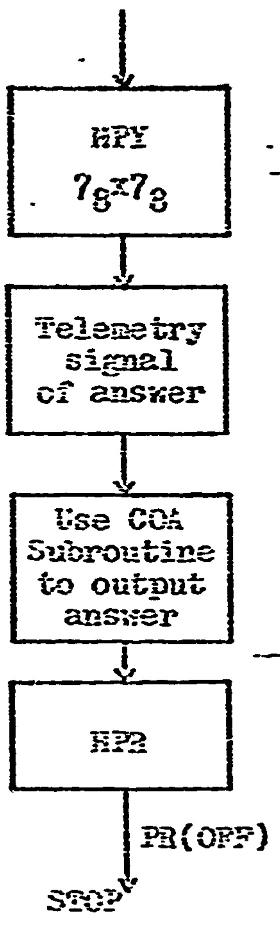
Program Address			Data Address	
Chan	Sect	Arithmetic Routine	Chan	Sect
00	000	ADD routine	02	001
04	000	SUB routine	06	001
10	000	SAD routine	12	001
14	000	SSU routine	16	001
20	000	MPY routine	22	001
24	000	SMP routine	26	001
30	000	Whole No. MPY routine	32	001

Each routine of the arithmetic program can link up with the COA subroutine to output the results. The COA subroutine is loaded in channels 44 and 46. The last instruction of the COA subroutine is an HPR instruction which puts the computer in the program halt mode.

In this example a flag store teleme[illegible] signal is also used to output the answer. No mode tracing or register display is used.

The flowchart for example program no. 3 is as follows:


START
PR(ON) MR(ON) PS(ON) EN(ON)
Load Program
Memory Dump
K(RUN) MR(ON)
ADD 60_8+17_8
Telemetry signal of answer
Use COA Subroutine to output answer
K(HALT)
Load more data
50003000 LOCATE COMPUTE K(RUN)

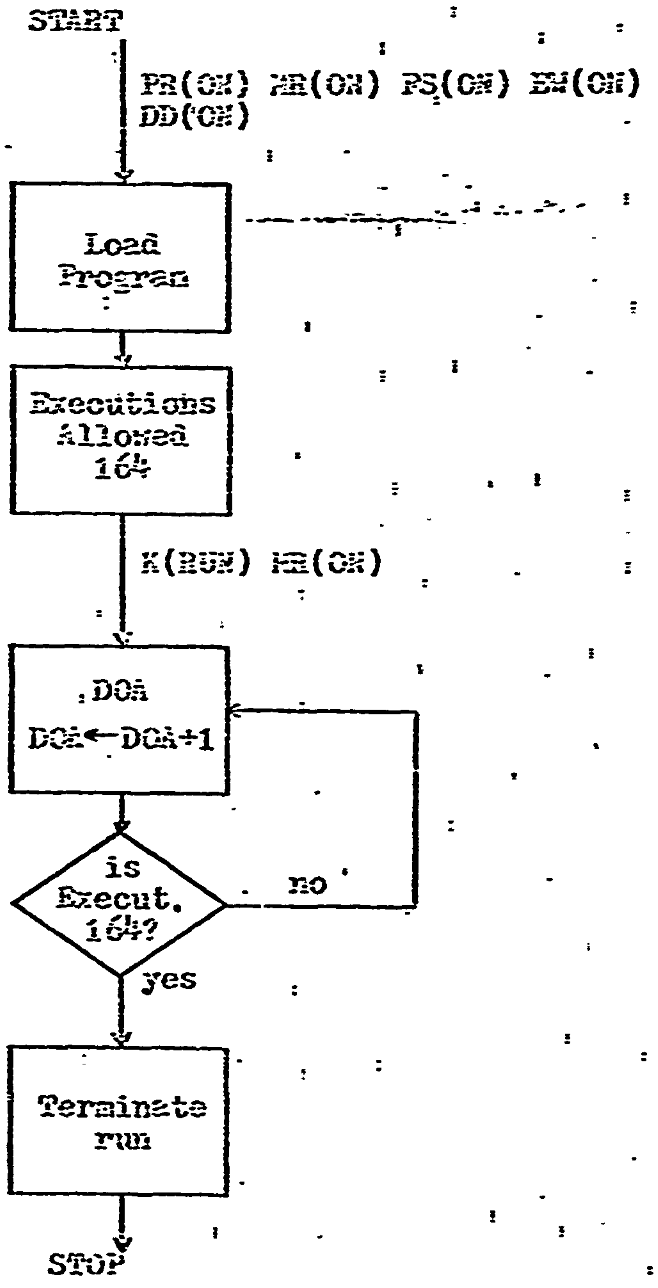

MPY
$7_8 x 7_8$
Telemetry signal of answer
Use COA Subroutine to output answer
HPR
PR(OFF)
STOP

```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
 WISE TYPE "N" - N

      ************************************************************
      **                                                        **
      **                    D17B COMPUTER                       **
      **                  SIMULATION PROGRAM                    **
      **                                                        **
      **    DATE=  01/24/72               TIME=  21.00.47       **
      **                                                        **
      ************************************************************


                  **  PRINTOUT OF INPUT PROGRAM  **
(ENTER PROGRAM)

S ARITHMETIC PROGRAM WITH CMA SUBROUTINE -

PR(ON) MR(ON) EX(ON) FF(ON) CHAN
Z 44010201 = 64020202 = 54034603 = 52404400 =
^ 400 9 44010601 = 74020602 = 54034603 = 52404400 =
^1000 9 44011201 = 60021202 = 54034603 = 52404400 =
^1400 9 44011601 = 70021602 = 54034603 = 52404400 =
^2000 9 44012201 = 24022202 = 54034603 = 52404400 =
^2400 9 44012601 = 20022602 = 54034603 = 52404400 =
^3000 9 44013201 = 00022214 = 47633202 = 00042213 =
         24056305 = 54034603 = 52404400 =

S CMA SUBROUTINE
^ 4400 9 44014601 = 47024602 = 40034000 = 00072205 = ^ 4407 9 54104610
= 44114601 = 03114201 = ^ 4415 9 47344614 = 40154200 = 03174601 = ^ 4417
9 64204605 = 54214613 = 44224602 = 10234400 = 74244624 = 54254624 =
44264626 = 74274627 = 11244400 = 44314631 = 54324624 = 40332200 = ^ 4601
9 01234567 = ^ 1 = ^ 4614 9 37777777 = ^ 4622 9 ^ 6 = ^ 4624 9 ^ 1 =
^4626 9 ^ 15 = ^ 1 = ^ 4631 9 ^ 6 =

S MEMORY CL 201 LOC CL 40 EX CL 17 EW EXEC(1000) MR(ON) Z(RUN, Z(HALT)
FILL CL 3201 LOC CL 7 EX CL 7 EW 53030000 LOC COMP Z(RUN)
PR(OFF)


                  **  RESULTS OF SIMULATION  **


INPUT SOURCE - CHAN TAPE


** MEMORY DUMP **

CHAN  SECT
 00   000       44010201       64020202       54034603       52404400
 04   000       44010601       74020602       54034603       52404400
 10   000       44011201       60021202       54034603       52404400
 14   000       44011601       70021602       54034603       52404400
 20   000       44012201       24022202       54034603       52404400
 24   000       44012601       20022602       54034603       52404400
 30   000       44013201       00022214       47633202       00042213
 30   004       24056305       54034603       52404400       77777777
```

```
44   000       44014601      47224602      40034200      00072205
44   004       77777777      77777777      77777777      54104610
44   010       44114601      00153201      77777777      47344614
44   014       40154200      00174021      77777777      64204606
44   020       54214605      44224622      10234430      74244624
44   024       54254624      44264626      74274627      10264400
44   030       44314631      54324624      40332200      77777777
46   000       77777777      01234567      00000001      77777777
46   014       37777777      77777777      77777777      77777777
46   020       77777777      77777777      00000006      77777777
46   024       00000001      77777777      00000015      00000001
46   030       77777777      00000006      77777777      77777777

** END OF MEMORY DUMP **     (PORTIONS OF MEMORY NOT LISTED CONTAIN NO
INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN)


NO. OF EXECUTIONS SPECIFIED = 1000
FLAGGED INSTRUCTION TELEMETRY SIGNAL - 000000 000000 000000 111111
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0111   HEXIDECIMAL CHARACTER OUTPUT - 7
BINARY CHARACTER OUTPUT - 0111   HEXIDECIMAL CHARACTER OUTPUT - 7
FLAGGED INSTRUCTION TELEMETRY SIGNAL - 000000 000000 000000 110001
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0000   HEXIDECIMAL CHARACTER OUTPUT - 0
BINARY CHARACTER OUTPUT - 0110   HEXIDECIMAL CHARACTER OUTPUT - 6
BINARY CHARACTER OUTPUT - 0001   HEXIDECIMAL CHARACTER OUTPUT - 1
TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT


    **  END OF PROGRAM          EXECUTION TIME=    1.721  SEC
21.13.29.STOP
```

Example Program Number 4. This example is a program which shows the discrete output capability of the simulation program. To interpret the discrete output listing, the reader should refer to Fig. 12 in Appendix C.

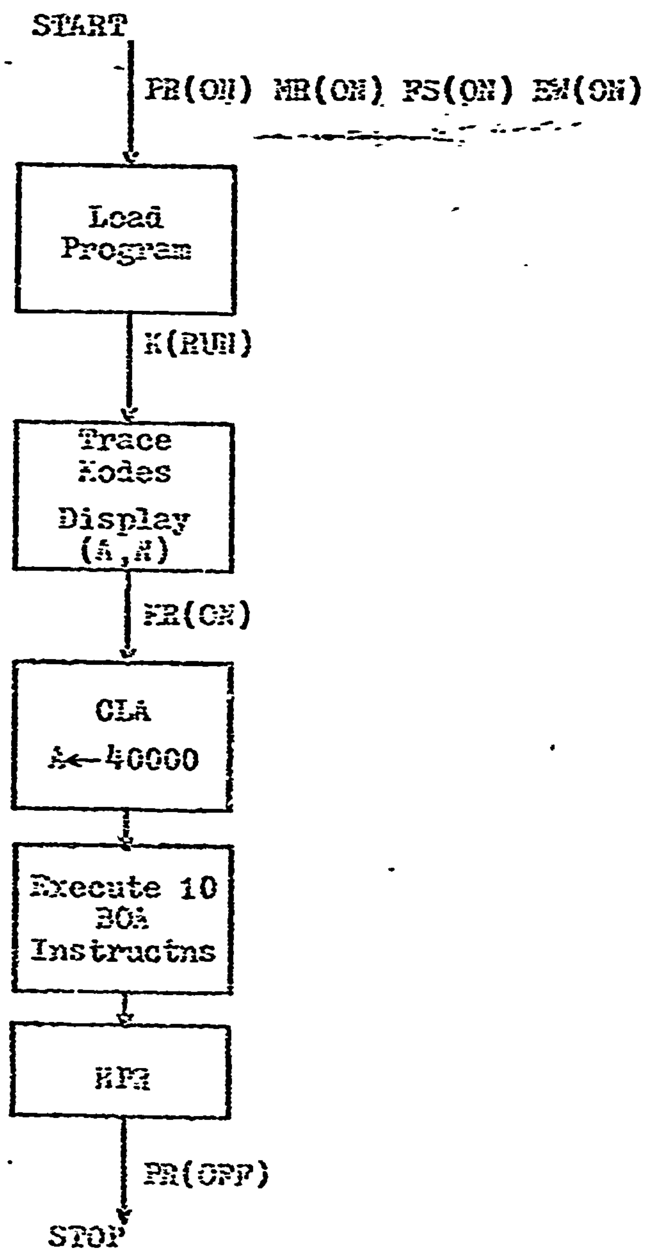
The flowchart for example program no. 4 is as follows:



```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
WISE TYPE "N" - N

          ****************************************************
          **                                                **
          **                  D17B COMPUTER                 **
          **               SIMULATION PROGRAM               **
          **                                                **
          **    DATE=  01/24/72              TIME=  19.16.37 **
          **                                                **
          ****************************************************

                    **  PRINTOUT OF INPUT PROGRAM  **
(ENTER PROGRAM)

$ DISCRETE OUTPUT PROGRAM

PR(ON) RE(ON) EN(ON) FILL
40012500 EN 44020500 EN 64030203 EN 54340302 EN 56000000 EN
CL 205 LOC CL 1 EN EXEC(0164) E(ON) RE(ON)
PR(OFF)


                    **  RESULTS OF SIMULATION  **

NO. OF EXECUTIONS SPECIFIED =  164
DISCRETE OUTPUT LINE D 10 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 20 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 30 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 40 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D [illegible] HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 10 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 40 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 20 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 40 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 30 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D [illegible] HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D 90 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D100 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D110 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D120 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D130 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D140 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D150 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D[illegible] HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D[illegible] HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D180 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D190 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D200 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D210 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D220 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D230 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D240 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D250 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D260 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D270 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D280 HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D[illegible] HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D[illegible] HAS A "1" OUTPUT SIGNAL
DISCRETE OUTPUT LINE D[illegible] HAS A "1" OUTPUT SIGNAL
NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
TO RUN ANOTHER PROGRAM TYPE "GO"; TO STOP TYPE "HALT" - HALT


     **  END OF PROGRAM          EXECUTION TIME=      .882  SEC
19.22.36.STOP :
```

Example Program Number 5. The program for this example uses the binary output instructions (BOA,BOB,BOC) and shows the binary output capability of the simulation program. To interpret the binary output listing, the reader should refer to Fig. 13 in Appendix C.

The flowchart for example program no. 5 is as follows:

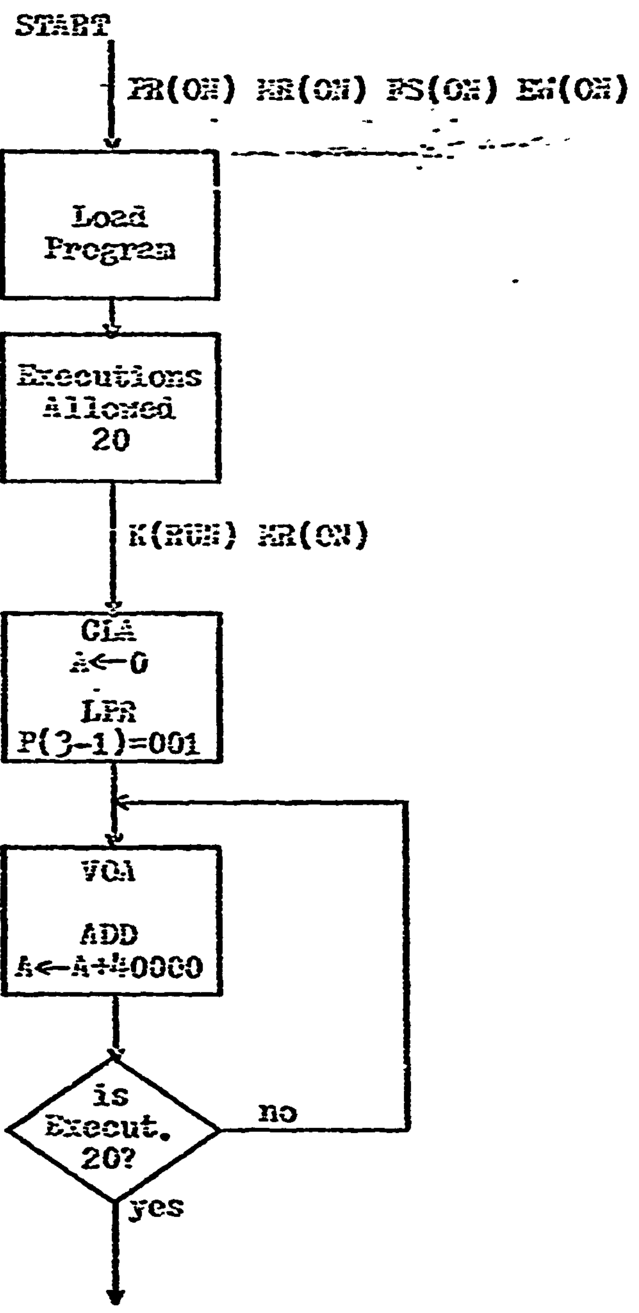

START
PR(ON) MR(ON) PS(ON) EM(ON)
Load Program
K(RUN)
Trace Modes Display (A,M)
PR(ON)
CLA
A←40000
Execute 10 BOA Instructns
HPR
PR(OFF)
STOP

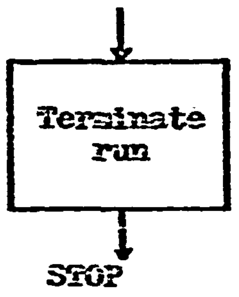
```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
 WISE TYPE "N" - N

     ****************************************************************
     **                                                            **
     **                      DITB COMPUTER                         **
     **                   SIMULATION PROGRAM                       **
     **                                                            **
     **    DATE=  01/24/72                     TIME=  20.10.14     **
     **                                                            **
     ****************************************************************


                   **  PRINTOUT OF INPUT PROGRAM  **
(ENTER PROGRAM)

 S BINARY OUTPUT PROGRAM

 PR(ON) MR(ON) FS(ON) EN(ON) FILL
 44010201 EN 40021000 EN 40031000 EN 40041000 EN 40051000 EN 40061000 EN
 40071000 EN 40101000 EN 40111000 EN 40121000 EN 40131000 EN 40002200 EN
 CL 201 LOC 00400000 EN M(RUN) SIGNAL REG(A,M) MR(ON)
 PR(OFF)


                   **  RESULTS OF SIMULATION  **

 SIGNAL ON - MODES WILL BE TRACED

 MASTER RESET SEQUENCE
 PREPARE TO OPERATE MODE
 SYNC BIT COUNTER 1 MODE
 SYNC BIT COUNTER 2 MODE

 S(4)*S(2)S(1)* IDLE SUB-MODE OF MANUAL HALT
 PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT

    :

 COMPUTE MODE

 TRANSFER INSTRUCTION - (TRA)

 M(24-1) = 000 000 100 000 000 000 000 000
 CLEAR & ADD INSTRUCTION - (CLA)
 A(24-1) = 000 000 100 000 000 000 000 000

 BINARY OUTPUT "A" INSTRUCTION - (BOA)
 BINARY OUTPUT ON LINE 010 OF +1
 A(24-1) = 000 000 010 000 000 000 000 000

 BINARY OUTPUT "A" INSTRUCTION - (BOA)
 BINARY OUTPUT ON LINE 010 OF +1
 A(24-1) = 000 000 000 000 000 000 000 000

 BINARY OUTPUT "A" INSTRUCTION - (BOA)
 BINARY OUTPUT ON LINE 011 OF -1
 A(24-1) = 111 111 110 000 000 000 000 000
```

```
BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 010 OF +1
A(24-1) = 000 000 000 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 011 OF -1
A(24-1) = 111 111 110 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 010 OF +1
A(24-1) = 000 000 000 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 011 OF -1
A(24-1) = 111 111 110 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 010 OF +1
A(24-1) = 000 000 000 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 011 OF -1
A(24-1) = 111 111 110 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (BOA)
BINARY OUTPUT ON LINE 010 OF +1
A(24-1) = 000 000 000 000 000 000 000 000

HALT AND PROCEED INSTRUCTION - (HPR)
PROGRAM HALT MODE

POWER HAS BEEN TURNED OFF
TO SEE ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT


     ** END OF PROGRAM          EXECUTION TIME=     .859  SEC
20.16.31.STOP
```

<u>Example Program Number 6</u>. This example is a program which uses the voltage output instructions (VOA,VOB,VOC) and shows the voltage output capability of the simulation program. To interpret the voltage output listing, the reader should refer to Fig. 14 in Appendix C.

The flowchart for this example is as follows:

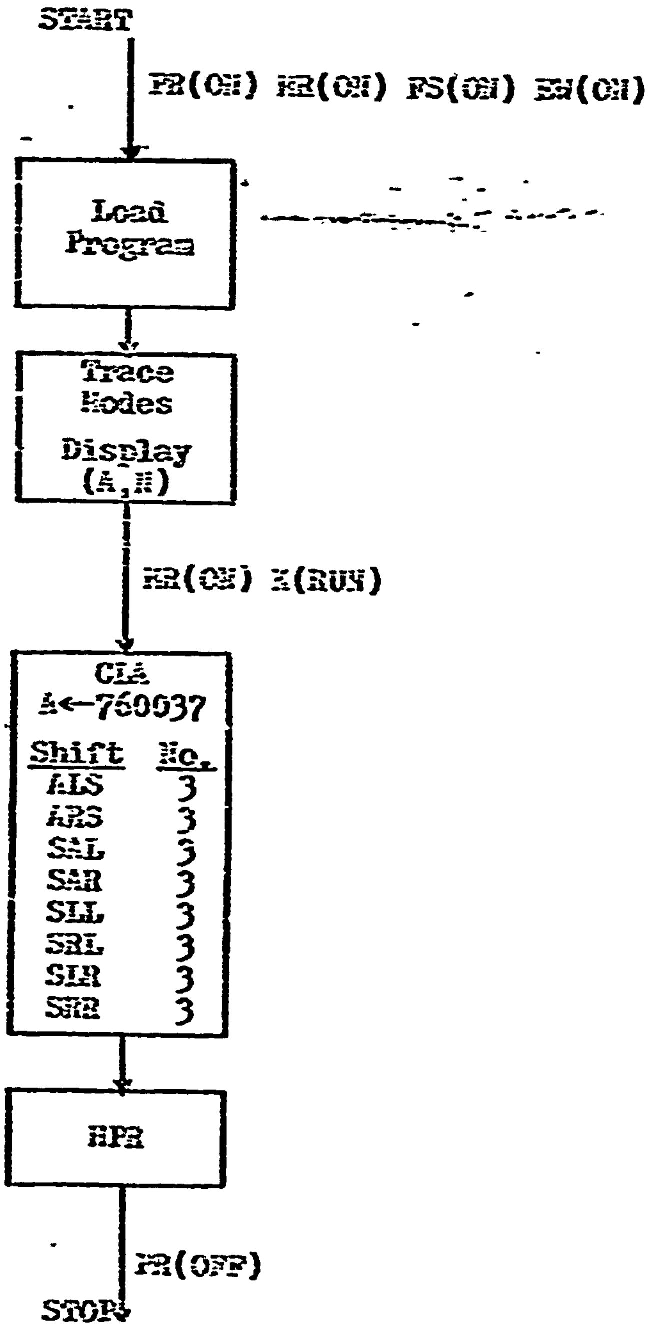



Terminate run
STOP

```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
WISE TYPE "N" - N

        ************************************************************
        **                                                        **
        **                     DITB COMPUTER                      **
        **                   SIMULATION PROGRAM                   **
        **                                                        **
        **    DATE=  02/07/72                TIME=  20.40.04      **
        **                                                        **
        ************************************************************


                 **  PRINTOUT OF INPUT PROGRAM  **
(ENTER PROGRAM)

$ VOLTAGE OUTPUT PROGRAM

PR(ON) HR(ON) FS(ON) EN(ON) FILL
40017200 EN 44020201 EN 40035000 EN 64020202 EN 50000002 EN
CL 201 LOC CL EN CL 200000 EN EXEC(0020) X(RUN) HR(ON)
PR(OFF)


                  **  RESULTS OF SIMULATION  **

NO. OF EXECUTIONS SPECIFIED =    20
PHASE REGISTER - P(8-1)= 001
VI(8-1)= 00000000 WITH A VOLTAGE OUTPUT OF   0.00 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000001 WITH A VOLTAGE OUTPUT OF    .16 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000010 WITH A VOLTAGE OUTPUT OF    .31 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000011 WITH A VOLTAGE OUTPUT OF    .47 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000100 WITH A VOLTAGE OUTPUT OF    .63 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000101 WITH A VOLTAGE OUTPUT OF    .78 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000110 WITH A VOLTAGE OUTPUT OF    .94 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1)= 00000111 WITH A VOLTAGE OUTPUT OF   1.09 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT


    **  END OF PROGRAM        EXECUTION TIME=      .771  SEC
20.45.04.STOP
```

<u>Example Program Number 7</u>. This program is an example of a program which uses all the shift instructions of the D17B computer instruction set. Mode tracing and register display are used in the compute mode only.

The flowchart for example program no. 7 is as follows:



```
IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHER-
 WISE TYPE "N" - N

      ****************************************************************
      **                                                            **
      **                      DIB COMPUTER                          **
      **                   SIMULATION PROGRAM                       **
      **                                                            **
      **   DATE=  01/24/72                  TIME=  19.36.25         **
      **                                                            **
      ****************************************************************

                    **  PRINTOUT OF INPUT PROGRAM  **
(ENTER PROGRAM)

$ SHIFT PROGRAM

PR(ON) MR(ON) PS(ON) SM(ON) FILL
44010201 EN 00022205 EN 00033204 EN 00042005 EN 00053003 EN
00062405 EN 00072605 EN 00103405 EN 00115505 EN 40002203 EN
CL 201 LOC 00760037 EN SIGNAL REGISTER(A,B) X(205) MR(ON)
PR(OFF)


                    **  RESULTS OF SIMULATION  **

SIGNAL MR - MODES WILL BE TRACED

MASTER RESET SEQUENCE
PREPARE TO OPERATE MODE
SYNC BIT COUNTER 1 MODE
SYNC BIT COUNTER 2 MODE

C(4)'0(2)0(1)' IDLE SUB-MODE OF MANUAL HALT
PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT

COMPUTE MODE

TRANSFER INSTRUCTION - (TRA)

B(24-1) = 000 000 111 110 000 000 011 111
CLEAR & ADD INSTRUCTION - (CLA)
A(24-1) = 000 000 111 110 000 000 011 111

ACCUMULATOR LEFT SHIFT INSTRUCTION - (ALS)
A(24-1) = 011 111 000 000 001 111 100 000

ACCUMULATOR RIGHT SHIFT INSTRUCTION - (ARS)
A(24-1) = 000 001 111 100 000 000 111 110

SPLIT ACCUMULATOR LEFT SHIFT INSTRUCTION - (SAL)
A(24-1) = 001 111 100 000 000 111 110 000

SPLIT ACCUMULATOR RIGHT SHIFT INSTRUCTION - (SAR)
A(24-1) = 000 001 111 100 000 000 111 110
```

```
SPLIT LEFT WORD LEFT SHIFT INSTRUCTION - (SLL)
A(24-1) = 001 111 100 000 000 000 111 110

SPLIT RIGHT WORD LEFT SHIFT INSTRUCTION - (SRL)
A(24-1) = 001 111 100 000 000 111 110 000

SPLIT LEFT WORD RIGHT SHIFT INSTRUCTION - (SLR)
A(24-1) = 000 001 111 100 000 111 110 000

SPLIT RIGHT WORD RIGHT SHIFT INSTRUCTION - (SRR)
A(24-1) = 000 001 111 100 000 000 111 110

HALT AND PROCEED INSTRUCTION - (HPR)
PROGRAM HALT MODE

POWER HAS BEEN TURNED OFF
TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT"


    **  END OF PROGRAM         EXECUTION TIME=     .317  SEC
19.41.59.STOP
```

VI. Conclusion

A software simulation program of ~~the Minuteman~~ D17B Computer was written to simulate the functions of the D17B computer. The objectives of this simulation were to have the simulation program simulate the actual computer as closely as possible. This objective was met because the majority of the D17B functions have been included in the simulation program. The loading and interaction functions of the noncompute mode have been used. In the compute mode, the searching, reading and writing memory, and instruction execution are all part of the simulation program. Wherever possible, the same algorithm implemented on the D17B was used in the simulation program. This approach resulted in some inefficiencies in the simulation program, but a by-product of using the same algorithm is that the simulation program can be used as a teaching aid for learning the operation of the D17B computer. Also error detection was built into the simulation program and has been very helpful in creating program tapes for the D17B computer.

<u>Recommendations for Future Study</u>. There are some D17B computer functions which have not been incorporated in the simulation program. Two of these functions are associated with real time control processing and include the capabilities for incremental inputs and fine count-down operations.

The D17B computer is capable of detecting and incrementally adding bits of information to the words of the V-loop and R-loop without program control. Also when the EFC (enter fine countdown) instruction is executed, the computer goes into the fine countdown mode. In the fine countdown mode, the V-loop and U-loop are linked together forming a digital integrator which operates without program control.

The incremental input and fine countdown mode functions listed above would be important if the simulation were to be used for real time control. Since this was not the original purpose of the simulation program, these functions are not included. However, the fine countdown instructions (HFC-halt fine countdown and EFC) and the instructions which store and unload information from the V-loop and R-loop are a part of the simulation program. Also, a subroutine for storing incremental data supplied by the user into the V-loop and R-loop is a part of the simulation program. The incremental input and fine countdown mode capabilities described above are improvements that could be added by a thesis student who is researching the area of real time control applications for the D17B computer.

Another recommendation for future study would be the creation of an assembler for the D17B computer. The assembler could be written for operation on the CDC 6600 computer. The assembler would accept as input a program written in D17B mnemonic coding and output on punched tape a machine language version of the program. This machine language program on the punched tape could then be supplied as data to both the D17B computer and the simulation program.

Bibliography

1. Beck, C. H. D17B Computer Programming Manual. Report HCUC-4-71. New Orleans, Louisiana: Tulane University Systems Laboratory, Department of Electrical Engineering, September 1971.

2. Chu, Y. Introduction to Computer Organization. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1970.

3. Control Data Corporation. Intercom 2 Reference Manual. Publication No. 60306000. Sunnyvale, California: CDC, 1970.

4. Hansen D. D. and Watkins H. R. A Rigorous Logical Study With Lab of the D17 Digital Computer. ACC-311702-33. Computers and Data Systems Logistics, Customer Training, 30 April 1962.

5. ——. Minuteman D17 Computer Training Data. Autonetics, a division of North American Rockwell, 8 June 1970.

6. ——. Preliminary Maintenance Manual of the Minuteman D17A Computer and Associated Test Equipment. Project Office Memorandum No. 71. Autonetics, a division of North American Aviation, Inc., January 1960.

7. Shroyer, L. O. D17A Computer Manual. Field Engineering (M^2 Logistics), 1 July 1960.

8. T. O. 1102-10-5-3-5. General Purpose Digital Computer (Model D17B). Technical Manual, Overhaul, Autonetics, 1 December 1963, changed 12 October 1964.

VITA

Bruce Chatterton was born on 31 March 1940 in Franklin, Idaho. He graduated from high school in Preston, Idaho in 1958 and attended Utah State University for five quarters. He enlisted in the USAF in December 1962 and received 36 weeks of training in electronics and communication equipment repair at Sheppard AFB, Texas. While stationed at McClellan AFB, California, he attended American River Junior College to become eligible for education under the Airman Education and Commission Program (AECP). He was accepted for AECP training in June 1965 and attended Oklahoma State University where he received the degree of Bachelor of Science in Electrical Engineering in July 1967. He then attended Officer Training School at Lackland AFB, Texas and received a commission in the USAF in November 1967. He served as a project officer for the Air Force Satellite Control Facility in Los Angeles, California. He attended the Air Force Institute of Technology where he received the degree of Master of Science in Electrical Engineering in March 1972.

Permanent Address: P.O. Box 177
Franklin, Idaho 83237

This thesis was typed by Bruce Chatterton

Appendix A

Printout of Simulation Program

```
      PROGRAM MSIM(INPUT,OUTPUT,TAPE7=INPUT,TAPE5=OUTPUT,TAPE5,TAPE4)
C
C***********************************************************************
C*                                                                    *
C*              [illegible] COMPUTER SIMULATION PROGRAM"               *
C*                                                                    *
C*    THIS VERSION IS WRITTEN IN THE FORTRAN EXTENDED LANGUAGE FOR     *
C*    USE ON THE INTERCOM SYSTEM OF A CDC 6600 COMPUTER SYSTEM         *
C*                                                                    *
C***********************************************************************
C
C
      INTEGER A(24), AX, C(5), CB(5), CHAR, CODE, CONREG(24,2), CF(5)
      INTEGER D(5), DX, FE, G(8), DY, F(4), FC, H(3), M(15), S(4), F(2)
      INTEGER MASK, N(24,4), DISPOSE, REGIST, REGISTR, RI, RX, SB(7)
      INTEGER SECT, B, Y(24,4), TI, TX, X(15,12), XI, XX, Y(24,30), YI
      INTEGER VI, HALT, OFF, ON, ONE, OUT, SINGLE, ZERO, MODE(2), FSSIG
      INTEGER ENTRY, FS, FSECT, FR, SIGNAL, T, TRAP, ISIG, ATIME, JC(3)
C
      DIMENSION VALUES(8), IF(5), M(3), MLIST(10)
C
      COMMON A, CHAR, CODE, CONREG, L2, E, EA, F, FC, M, I(24), IREG
      COMMON L(24), M(128,20), R(24), ECOL, FLAG, NOP, WORD(72), PHASE
      COMMON SB, SIGNAL, Z, REGIST, REGISTR, RI, RX, SECT, L, V, VI, VX
      COMMON X, XI, Y, YI, DISPOSE
C
      COMMON/MDATA/ ON, ZERO, ONE, AREG(12), MC(27), MELAPS, MCOUNT,
     1              MLAPSE, NSFIRST, ATIES
C
      DATA MODE/ 1H3, 1H4, 1H2, 1HF, 1HC, 1H5, 1HE, 1H7/
      DATA MLIST/1H0, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9/
      DATA VALUES/.078125,.15625,.3125,.625,1.25,2.5,5.,10./
      DATA M/1H0, 1H3, 1H2, 1H5, 1H4, 1H6, 1H9, 1H3/
      DATA OFF/3HOFF/, SINGLE/6HSINGLE/, HALT/4HHALT/, RUN/3HRUN/
C
C     ****************************************************************
C     *               READING & TRANSLATION SECTION                 *
C     *        OF THE [illegible] COMPUTER SIMULATION PROGRAM        *
C     ****************************************************************
C
COMMENT     **LABELING OUTPUT AND INITIALIZING VARIABLES
      WRITE(4,2020)
      WRITE(5,2020)
      READ(7,2020) DISPOSE,(MODE(I1),I1=1,3)
      I2=DATE(I1)
      I3=TIME(I1)
      IF(DISPOSE.NE.MA(1P)) WRITE(5,2030) I2, I3
      IF(DISPOSE.EQ.MA(1E)) WRITE(5,2040) I2, I3, (MODE(I1),I1=1,2)
    1 DO 5 I2=1,21
      DO 5 I1=1,125
    5 M(I1,I2)=VALUES
      DO 10 I1=1,5
   10 S(I1)=0
      XI=YI=DX=FC=FE=AS=OPR=0
      GO TO 20
   15 WRITE(5,2025)
      WRITE(4,2025)
      READ(7,2010) MODE(1)
      IF(MODE(1).NE.RUN) GO TO 425
   20 WRITE(5,2015)
      WRITE(4,2015)
      LIST=50
      PHASE=ISEC=T=OFF
      ENTRY=ISIG=ISECT=XI=YI=IMSIG=FSSIG=IFLG=AX=TX=RSTFG=MASK=0
   25 SIGNAL=FLISTR=FLAG=LIST1=RI=VI=ISIG=0
      K=HALT
C
COMMENT     **ACCEPTING INPUT
      REWIND 5
   30 READ(7,2000)(WORD(I1),I1=1,8)
      WRITE(5,2020)(WORD(I1),I1=1,8)
      IF(DISPOSE.EQ.MA(1E)) WRITE(5,2020)(WORD(I1),I1=1,8)
      IF(WORD(1).EQ.10HEOF(OFF)    ) GO TO 35
      IF(WORD(1).EQ.10HEO          ) GO TO 35
      GO TO 30
   35 REWIND 5
      WRITE(5,2025)
      WRITE(4,2025)
C
COMMENT     **READING & TRANSLATION OF INPUT DATA
   40 NCOL=0
      READ(5,2000)(WORD(I1),I1=1,72)
      IF(WORD(1).EQ.1H ) GO TO 45
      IF([illegible](1).[illegible]) GO TO 45
      WRITE(6,2035)
      WRITE(4,2035)
      GO TO 25
   45 IF([illegible].EQ.1) GO TO 1[illegible]
      NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 40
      IF(WORD(NCOL).EQ.BLANK) GO TO 45
      DO 50 I1=1,25
      IF(WORD(NCOL).EQ.[illegible](I1)) GO TO (125,125,125,125,125,125,125,125,
     1 135,135,70,75,135,85,95,[illegible],[illegible],355,[illegible],75,225,230,240,85,270,
     2 275) I1
   50 CONTINUE
   55 DO 60 I1=NCOL,72
      IF(WORD(I1).EQ.BLANK) GO TO 65
   60 CONTINUE
   65 WRITE(6,2040)(WORD(I2),I2=NCOL,I1)
      WRITE(4,2040)(WORD(I2),I2=NCOL,I1)
      NCOL=I1
      GO TO 45
   70 IF(WORD(NCOL+1).EQ.[illegible](19)) GO TO 135
      IF(WORD(NCOL+1).EQ.[illegible](23)) GO TO 240
      GO TO 55
   75 IF(WORD(NCOL+1).EQ.[illegible](14)) GO TO 265
      IF(WORD(NCOL+1).EQ.[illegible](24)) GO TO 510
      IF(WORD(NCOL+1).EQ.[illegible](22)) GO TO 250
      GO TO 55
   85 IF(WORD(NCOL+1).EQ.[illegible](4)) GO TO 135
      IF(WORD(NCOL+1).EQ.[illegible](27)) GO TO 250
      IF(WORD(NCOL+1).EQ.[illegible](25)) GO TO 255
      GO TO 55
   95 IF(WORD(NCOL+2).EQ.[illegible](3)) GO TO 421
      IF(WORD(NCOL+1).EQ.[illegible](14)) GO TO 260
      IF(WORD(NCOL+1).EQ.[illegible]) GO TO 280
      IF(WORD(NCOL+1).EQ.[illegible](17)) GO TO 220
      GO TO 55
   9[illegible] IF(WORD(NCOL+1).EQ.[illegible](14)) GO TO 135
      IF(WORD(NCOL+1).EQ.[illegible]) GO TO 285
      IF(WORD(NCOL+1).EQ.[illegible](27)) GO TO 330
      GO TO 55
   95 IF(WORD(NCOL+1).EQ.[illegible](14)) GO TO 135
      IF(WORD(NCOL+1).EQ.[illegible](15)) GO TO 325
      IF(WORD(NCOL+1).EQ.[illegible](24)) GO TO 290
      GO TO 55
C
COMMENT      **INTERPRETATION OF [illegible] [illegible] DATA
  100 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 40
      IF(WORD(NCOL).EQ.BLANK) GO TO 100
      DO 105 I1=1,8
      IF(WORD(NCOL).EQ.[illegible](I1)) GO TO 125
  105 CONTINUE
      DO 110 I1=1,8
      IF(WORD(NCOL).EQ.[illegible](I1)) GO TO 115
  110 CONTINUE
      IF(WORD(NCOL).EQ.[illegible](25)) GO TO 255
      WRITE(6,2045) WORD(NCOL)
      WRITE(4,2045) WORD(NCOL)
      GO TO 100
  115 DO 120 I1=1,5
  120 IT(I1)=0
      IT(4)=1
      IF(WORD(NCOL).EQ.[illegible](2).OR.[illegible](NCOL).EQ.[illegible](7)) IT(5)=1
      IF(WORD(NCOL).EQ.[illegible](5).OR.[illegible](NCOL).EQ.[illegible](8)) IT(5)=1
      IF(WORD(NCOL).EQ.[illegible](5).OR.[illegible](NCOL).EQ.[illegible](6)) IT(3)=1
      IF(WORD(NCOL).EQ.[illegible](7).OR.[illegible](NCOL).EQ.[illegible](8)) IT(3)=IT(2)=1
      IF(WORD(NCOL).EQ.[illegible](7).OR.[illegible](NCOL).EQ.[illegible](4)) IT(2)=1
      IF(WORD(NCOL).EQ.[illegible](7).OR.[illegible](NCOL).EQ.[illegible](4)) IT(1)=1
      IF(WORD(NCOL).EQ.[illegible](8).OR.[illegible](NCOL).EQ.[illegible](8)) IT(1)=1
      GO TO 135
C
COMMENT      **TRANSLATION OF OCTAL DATA
  125 IF([illegible].EQ.1) GO TO 155
      DO 130 I1=1,5
  130 IT(I1)=0
      IF(WORD(NCOL).EQ.[illegible](1).OR.[illegible](NCOL).EQ.[illegible](6)) IT(5)=1
      IF(WORD(NCOL).EQ.[illegible](2).OR.[illegible](NCOL).EQ.[illegible](7)) IT(5)=1
      IF(WORD(NCOL).EQ.[illegible](5).OR.[illegible](NCOL).EQ.[illegible](6)) IT(3)=1
      IF(WORD(NCOL).EQ.[illegible](7).OR.[illegible](NCOL).EQ.[illegible](8)) IT(3)=IT(2)=1
      IF(WORD(NCOL).EQ.[illegible](7).OR.[illegible](NCOL).EQ.[illegible](4)) IT(2)=1
      IF(WORD(NCOL).EQ.[illegible](7).OR.[illegible](NCOL).EQ.[illegible](4)) IT(1)=1
      IF(WORD(NCOL).EQ.[illegible](5).OR.[illegible](NCOL).EQ.[illegible](8)) IT(1)=1
      GO TO 135
C
COMMENT     **TRANSLATION OF LOAD CODES
  135 IF(PRINT.EQ.1) GO TO 150
      DO 140 I1=1,5
  140 IT(I1)=0
      IT(4)=1
      IF(INPUT(NCOL).EQ.LN(10).OR.INPUT(NCOL).EQ.NN(11))       IT(5)=1
      IF(INPUT(NCOL).EQ.LN(15))                                 IT(5)=1
      IF(INPUT(NCOL+1).EQ.NP(21).AND.INPUT(NCOL).EQ.NN(13))    IT(5)=1
      IF(INPUT(NCOL).EQ.NN(13).OR.INPUT(NCOL).EQ.NN(14))       IT(2)=1
      IF(INPUT(NCOL).EQ.LN(15))                          IT(2)=IT(3)=1
      IF(INPUT(NCOL).EQ.LN(10).OR.INPUT(NCOL).EQ.NN(12))       IT(2)=1
      IF(INPUT(NCOL).EQ.LN(10).AND.INPUT(NCOL+1).EQ.NN(13))    IT(2)=1
      IF(INPUT(NCOL).EQ.LN(10).OR.INPUT(NCOL).EQ.NN(12))       IT(1)=1
      IF(INPUT(NCOL).EQ.LN(14).OR.INPUT(NCOL).EQ.NN(15))       IT(1)=1
  145 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 195
      IF(INPUT(NCOL).EQ.BLANK) GO TO 145
      GO TO 145
  150 WRITE(6,2045)
      WRITE(4,2045)
      GO TO 155
C
COMMENT     **TRANSLATION OF BINARY INPUT DATA
  155 I2=NCOL
      DO 160 I1=1,6
      IF(INPUT(I2).NE.NN(1).AND.INPUT(I2).NE.NN(2)) GO TO 165
      IT(6-I1)=0
      IF(INPUT(I2).EQ.NN(2)) IT(5-I1)=1
      I2=I2+1
  160 CONTINUE
      IF(INPUT(NCOL+5).NE.BLANK) GO TO 165
      NCOL=NCOL+5
      GO TO 65
  165 DO 170 I1=NCOL,72
      IF(INPUT(I1).EQ.BLANK) GO TO 175
  170 CONTINUE
  175 WRITE(5,2050)(INPUT(I2),I2=NCOL,I1)
      WRITE(4,2050)(INPUT(I2),I2=NCOL,I1)
      NCOL=I1
      GO TO 65
C
COMMENT     **TRANSLATION OF TIMING SIGNAL
  180 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 185
      IF(INPUT(NCOL).EQ.BLANK) GO TO 185
      GO TO 190
  185 T=0
      IF(IT.EQ.0) GO TO 190
      IF(ISIG.EQ.1) GO TO 560
      IF(ISIG.EQ.1) GO TO 515
      WRITE(5,2055)
      WRITE(4,2055)
      GO TO 15
  190 WRITE(6,2065)
      WRITE(4,2065)
      GO TO 65
C
COMMENT     **TRANSLATION OF COMPUTE MODE SWITCH
  195 IF(INPUT(NCOL+2).EQ.NN(23)) GO TO 200
      IF(INPUT(NCOL+2).EQ.NN(19)) GO TO 205
      IF(INPUT(NCOL+2).EQ.NN(24)) GO TO 210
      WRITE(6,2065)
      GO TO 165
  200 K=SINGLE
      GO TO 215
  205 K=MULT
      GO TO 215
  210 K=POS
  215 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 220
      IF(INPUT(NCOL).EQ.BLANK) GO TO 220
      GO TO 215
  220 IF(ISIG.EQ.1.AND.K.EQ.POS) GO TO 420
      IF(ISIG.EQ.1.AND.K.NE.MULT) GO TO 575
      IF(ISIG.EQ.2.AND.K.NE.MULT) GO TO 630
      IF(ISIG.EQ.1.AND.K.EQ.MULT) GO TO 365
      GO TO 65
C
COMMENT     **[illegible] SECTION
  225 NCOL=NCOL+1
```

```
      IF(NCOL.GT.72) GO TO 40
      IF(INPUT(NCOL).EQ.BLANK) GO TO 45
      GO TO 225
C
COMMENT      **TRANSLATION OF BINARY/OCTAL DESIGNATION
  230 BINARY=1
      GO TO 225
  235 BINARY=8
      GO TO 225
C
COMMENT      **TRANSLATION OF SIGNAL COMMAND
  240 IF(SIGNAL.EQ.1) GO TO 245
      SIGNAL=1
      WRITE(6,2270)
      GO TO 225
  245 SIGNAL=0
      WRITE(6,2275)
      GO TO 225
C
COMMENT      **TRANSLATION OF MASK CODE DESIGNATION
  250 MASK=1
      WRITE(6,2280)
      GO TO 225
  255 MASK=0
      GO TO 225
C
COMMENT      **TRANSLATION OF REGISTER DISPLAY COMMAND
  260 CALL REG1
      GO TO 45
C
COMMENT      **TRANSLATION OF MEMORY DISPLAY COMMAND
  265 CALL MEMORY
      GO TO 45
C
COMMENT      **TRANSLATION OF INCREMENTAL & DISCRETE INPUTS
  270 CALL DIS X
      GO TO 45
  275 CALL DIS Y
      GO TO 45
  280 CALL INC R
      GO TO 45
  285 CALL INC V
      GO TO 45
C
COMMENT      **TRANSLATION OF DISCRETE SIGNAL FLIPFLOP
  290 DF=1
      GO TO 225
C
COMMENT      **TRANSLATION OF EXECUTE SPECIFICATION
  295 CONTINUE
  300 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 40
      IF(INPUT(NCOL).EQ.ALPHEQ) GO TO 305
      GO TO 300
  305 DO 315 I1=1,4
      DO 310 I2=1,10
      IF(INPUT(NCOL+I1).EQ.ALIST(I2)) GO TO 315
  310 CONTINUE
      WRITE(5,2390)
      WRITE(6,2390)
      GO TO 225
  315 COMREG(I1)=I2-1
      LIST=1000*COMREG(1)+100*COMREG(2)+10*COMREG(3)+COMREG(4)
      WRITE(5,2395) LIST
      WRITE(6,2395) LIST
      GO TO 225
C
COMMENT      **TRANSLATION OF PR & PC FLIPFLOP SETTINGS
  320 IF(INPUT(NCOL+3).EQ.AN(2)) GO TO 325
      PR=0
      GO TO 225
  325 PR=1
      GO TO 225
  330 IF(INPUT(NCOL+3).EQ.AN(2)) GO TO 335
      PC=0
      GO TO 225
  335 PC=1
      GO TO 225
C
COMMENT      **TRANSLATION OF FILL SWITCH
  340 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 345
      IF(INPUT(NCOL).EQ.BLANK) GO TO 345
      GO TO 349
  345 FS=ON
      IF(FSSIS.EQ.1) GO TO 540
      IF(FSSIS.EQ.2) GO TO 570
      GO TO 45
C
COMMENT      **TRANSLATION OF COLD-STORAGE WRITE SWITCH
  350 IF(WORD(NCOL+4).EQ.YES(4)) GO TO 355
      IF(WORD(NCOL+4).EQ.YES(E)) GO TO 360
      WRITE(6,2130)
      GO TO 165
  355 EW=ON
      GO TO 225
  360 EW=OFF
      GO TO 225
C
COMMENT      **TRANSLATION OF DISCRETE SWITCH
  365 IF(WORD(NCOL+4).EQ.YES(4)) GO TO 370
      IF(WORD(NCOL+4).EQ.YES(E)) GO TO 375
      WRITE(6,2135)
      GO TO 165
  370 DS=ON
      GO TO 225
  375 DS=OFF
      GO TO 225
C
COMMENT      *TRANSLATION OF MECHANICAL INPUT SWITCH
  380 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 395
      IF(WORD(NCOL).EQ.BLANK) GO TO 385
      GO TO 380
  385 IM=ON
      IF(IMSIG.EQ.1) GO TO 540
      GO TO 45
C
COMMENT      **POWER-ON/OFF SEQUENCE
  390 IF(WORD(NCOL+4).EQ.YES(4)) GO TO 400
      IF(WORD(NCOL+4).EQ.YES(E)) GO TO 395
      WRITE(6,2110)
      GO TO 165
  395 PW=OFF
      IF(SIGNAL.EQ.1) WRITE(6,2115)
      WRITE(4,2115)
      GO TO 15
  400 PW=ON
      IF(SIGNAL.EQ.1) WRITE(6,2120)
      GO TO 225
C
  405 WRITE(6,2125)
      WRITE(4,2125)
      GO TO 15
  410 WRITE(6,2130)
      WRITE(4,2130)
      GO TO 15
  415 WRITE(6,2135)
      WRITE(4,2135)
      GO TO 15
  420 WRITE(6,2145)
      WRITE(4,2145)
      GO TO 15
  425 WRITE(6,2150)
      WRITE(4,2150)
      GO TO 1
  435 VOLTAGE=SECOND(VOLTAGE)
      WRITE(6,2140) VOLTAGE
      WRITE(4,2140) VOLTAGE
      STOP
C
C
C     ****************************************************************
C     *                  AUTOMATIC MODE SECTION                      *
C     *        OF THE [illegible] COMPUTER SIMULATION PROGRAM         *
C     ****************************************************************
C
COMMENT      **MASTER RESET SEQUENCE
  500 IF(MR.EQ.OFF) GO TO 410
  505 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 515
      IF(WORD(NCOL).EQ.BLANK) GO TO 515
      GO TO 505
  515 MR=ON
      IF(SIGNAL.EQ.1) WRITE(6,2200)
C     CONTINUE
```

```
C      RC=1
C
COMMENT       **PREPARE TO OPERATE MODE
       IF(SIGNAL.EQ.1) WRITE(6,3005)
       P(3)=P(2)=P(1)=0
       FC=0
       D(5)=D(4)=D(3)=D(2)=D(1)=0
       SP(2)=3
       SP(3)=1
C      D=1
C      E=5
C      D(2)=1
C      D(4)=0
C      D(2)=0
       TS=OFF
C      J=1
C
COMMENT       **SYNC BIT COUNTER 1
       IF(SIGNAL.EQ.1) WRITE(6,3010)
C      O(1)=1
C
COMMENT       **SYNC BIT COUNTER 2
       IF(SIGNAL.EQ.1) WRITE(6,3015)
       OPP=6
       DO 515 I1=1,24
  515 I(I1)=ZERO
  :    I(22)=I(24)=ONE
       IF(REGISTR.EQ.3) GO TO 520
       REGIST=RREG(3)
       CALL REG2
C      O(1)=0
C      RC=8
C
COMMENT       **D(4)*D(2)D(3)' IDLE SUB-MODE OF MANUAL HALT
  525 IF(SIGNAL.EQ.1) WRITE(6,3020)
       IF(SP(2).EQ.3.AND.K.NE.HALT) GO TO 535
C      O(4)=1
       GO TO 535
  530 IF(SIGNAL.EQ.1) WRITE(6,3025)
       IF(SP(2).EQ.3.AND.K.NE.HALT) GO TO 533
C      D(1)=1
       GO TO 573
  533 CONTINUE
C      D(2)=0
       GO TO 533
C
COMMENT       **INTERLOCK SUB-MODE OF MANUAL HALT
  535 IF(IF.EQ.ON.OR.SP(2).EQ.3.AND.K.NE.HALT) GO TO 550
C      D(1)=1
       IF(SIGNAL.EQ.1) WRITE(6,3030)
  540 IF(FS.EQ.ON) GO TO 545
       IF(IF.EQ.ON.OR.SP(2).EQ.3.AND.K.NE.HALT) GO TO 555
C      O(2)=1
       IFSIG=FSSIG=1
       GO TO 563
  545 CONTINUE
C      RC=1
       GO TO 625
  550 IF(SIGNAL.EQ.1) WRITE(6,3035)
C      C(4)=0
       GO TO 525
  555 FSSIG=2
       IFSIG=2
C      O(4)=0
       GO TO 573
C
COMMENT       **PREPARE TO LOAD SUB-MODE OF MANUAL HALT
  560 IF(SIGNAL.EQ.1) WRITE(6,3035)
       IF(T.NE.ON) GO TO 565
C      RC=1
       GO TO 535
  565 IF(SIGNAL.EQ.1) WRITE(6,3040)
C      D(2)=1
       CSINC=1
       GO TO 65
C
COMMENT       **D(4)*D(2)D(3) IDLE SUB-MODE OF MANUAL HALT
  570 IF(FS.EQ.ON) GO TO 533
  575 IF(SIGNAL.EQ.1) WRITE(6,3045)
       IF(SP(3).EQ.3) SP(2)=1
       IF(SP(2).EQ.3.AND.K.NE.HALT) GO TO 555
C      D(1)=0
```

```
      IF(IM.EQ.OFF) GO TO 500
      IF(SIGNAL.EQ.1) WRITE(6,3450)
      FSSIG=2
      GO TO 45
  500 IF(SIGNAL.EQ.1) WRITE(6,3545)
C     YC=1
      GO TO 635
  505 CONTINUE
C     D(2)=0
C
COMMENT      **PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT
  510 IF(SIGNAL.EQ.1) WRITE(6,3555)
C     MD=1
C     D=0
C     K=1
      FSECT=NTIME=NSIG=NSIMS=FSSIG=IMSIG=TSIG=0
      GO TO 130
C
COMMENT      **WAIT MODE
  515 TSIG=1
      SP(2)=FSSIG=IMSIG=NSIMS=0
      FS=IM=OFF
  520 IF(SIGNAL.EQ.1) WRITE(6,3060)
  525 CP(4)=CP(3)=CP(2)=CP(1)=ZERO
C     CP(5)=1
      IF(T.EQ.ON) GO TO 623
C     RC=0
      IF(SP(2).EQ.1) GO TO 625
C     D(4)=1
      GO TO 45
  623 CONTINUE
C     RC=1
      GO TO 630
  625 WRITE(6,3165)
C     YC=0
      TSIG=0
      GO TO 575
C
COMMENT      **PREPARE TO SAMPLE MODE
  570 IF(SIGNAL.EQ.1) WRITE(6,3070)
C     J=0
C
COMMENT      **SAMPLE CODE
      IF(SIGNAL.EQ.1) WRITE(6,3075)
C     CP(5)=0
C     D(4)=0
      DO 635 I1=1,4
      IF(IT(I1).EQ.1) CP(I1)=ONE
  635 CONTINUE
      SP(3)=IT(5)
      T=OFF
C     RC=3
C
COMMENT      **PARITY CHECK
      IF(SIGNAL.EQ.1) WRITE(6,3030)
C     D(4)=1
      DO 645 I1=1,4
      IF(SP(3).EQ.0.AND.CP(I1).EQ.ONE) GO TO 640
      IF(SP(3).EQ.1.AND.CP(I1).EQ.ONE) SP(3)=0
      GO TO 645
  640 SP(3)=1
  645 CONTINUE
C     D(4)=0
C     CP(5)=1
C
COMMENT      **PROCESS CODE-GENERAL
      IF(CP(4).EQ.ZERO) GO TO 655
      CODE=4*CP(3)+2*CP(2)+CP(1)+1
      GO TO (670,675,535,530,555,710,770,785) CODE
C
COMMENT      **PROCESS CODE-OCTAL
  655 IF(SIGNAL.EQ.1) WRITE(6,3595)
C     D(4)=1
      DO 660 I2=1,21
  660 L(25-I2)=L(22-I2)
      DO 665 I1=1,3
  665 L(I1)=CP(I1)
      IF(REGISTR.EQ.5) GO TO 785
      REGIST=NREG(2)
      CALL REG2
C     D(4)=0
      GO TO 735
C
```

```
COMMENT      **PROCESS CODE-HALT
  570 IF(SIGNAL.EQ.1) WRITE(6,3090)
C       NC=8
        GO TO 795
C
COMMENT      **PROCESS CODE-LOCATION
  575 IF(SIGNAL.EQ.1) WRITE(6,3095)
        KMPR=0
        DO 680 I1=1,24
  680 I(I1)=L(I1)
        IF(REGIST2.EQ.0) GO TO 795
        REGIST=NREG(3)
        CALL REG2
        IF(SIGNAL.EQ.1) WRITE(6,4641)
        GO TO 735
C
COMMENT      **PROCESS CODE-FILL
  595 IF(SIGNAL.EQ.1) WRITE(6,3100)
        D(2)=0
        GO TO 735
C
COMMENT      **PROCESS CODE-VERIFY
  600 IF(SIGNAL.EQ.1) WRITE(6,3105)
        D(2)=1
        GO TO 735
C
COMMENT      **PROCESS CODE-START TO COMPUTE
  635 IF(SIGNAL.EQ.1) WRITE(6,3110)
C       R(2)=1
C       NC=9
C       J=1
        ISIG=2
        GO TO 575
C
COMMENT      **PROCESS CODE-ENTER
  700 IF(SIGNAL.EQ.1) WRITE(6,3115)
C
COMMENT      **ENTER-IDLE SUBMODE OF FILL-VERIFY
        IF(SIGNAL.EQ.1) WRITE(6,3120)
        DO 705 I1=1,24
  705 I(I1)=L(I1)
C       J=5
        IF(REGIST2.EQ.0) GO TO 710
        REGIST=NREG(1)
        CALL REG2
  710 IF(SP(3).NE.1) GO TO 715
C       J=1
        GO TO 790
C
COMMENT      **ENTER-SWEEP SEARCH SUBMODE OF FILL-VERIFY
  715 IF(SIGNAL.EQ.1) WRITE(6,3125)
        SECT=64*I(7)+32*I(6)+16*I(5)+8*I(4)+4*I(3)+2*I(2)+I(1)+1
C       NC=2
        DO 720 I1=1,5
  720 CP(I1)=I(I1+7)
C       D=1
C       E=1
C
COMMENT      **ENTER-WAIT 2 S.T. SUBMODE OF FILL-VERIFY
        IF(SIGNAL.EQ.1) WRITE(6,3130)
        DO 725 I1=1,5
  725 C(I1)=CP(I1)
        CHAN=16*C(5)+8*C(4)+4*C(3)+2*C(2)+C(1)+1
        IF(CHAN.GT.24) GO TO 455
        IF(C(3).EQ.1) GO TO 730
        IF(CHAN.GE.22.AND.(CHAN.NE.23.AND.CHAN.NE.24)) GO TO 735
        CALL LOAD
        IF(XFLAG.EQ.1) GO TO 15
        GO TO 735
  730 CALL UNLOAD
        IF(XFLAG.EQ.1) GO TO 15
  735 CONTINUE
C       NC=1
C       D=1
C
COMMENT      **EXECUTE SUBMODE OF FILL-VERIFY
        IF(SIGNAL.EQ.1) WRITE(6,3135)
        DO 745 I1=1,23
        NUM=I(I1)+ONE
        IF(NUM.NE.2) GO TO 745
  740 I(I1)=ZERO
  745 I(I1)=ONE
```

```
      IF(REGISTR.EQ.3) GO TO 795
      REGIST=NREG(3)
      CALL REG2
  750 IF(Q(3).EQ.1) GO TO 755
      IF(CHAR.EQ.22.OR.CHAR.EQ.23.OR.CHAR.EQ.24) GO TO 735
      CALL LOAD
      IF(SFLAG.EQ.1) GO TO 25
      GO TO 795
  755 DO 760 I1=1,24
      IF(A(I1).EQ.X(I1)) GO TO 760
      SP(2)=1
      GO TO 765
  760 CONTINUE
  765 IF(SIGNAL.EQ.1.AND.REGISTR.EQ.1) WRITE(6,4341)
C     D=0
C     K=3
      GO TO 785
C
COMMENT      **PROCESS CODE-CLEAR
  770 IF(SIGNAL.EQ.1) WRITE(6,3140)
      DO 775 I1=1,24
  775 L(I1)=ZERO
      IF(REGISTR.EQ.3) GO TO 795
      REGIST=NREG(2)
      CALL REG2
      GO TO 795
C
COMMENT      **PROCESS CODE-DELETE
  780 IF(SIGNAL.EQ.1) WRITE(6,3145)
C
COMMENT      **PROCESS CODE-GENERAL
  785 CONTINUE
C     D(2)=1
C     J=1
      IF(SP(3).EQ.0) GO TO 790
      GO TO 615
  790 WRITE(6,4150)
C     NC=8
      GO TO 575
C
COMMENT      **PROCESS HALT
  795 IF(SIGNAL.EQ.1) WRITE(6,3155)
C     D=1
C     K=3
      IF(K.EQ.RUN) GO TO 45
  800 SP(2)=0
      SP(3)=1
C     D(2)=1
C     D(4)=1
C     J=1
      ISIG=KSIG=0
      IF(C(1).EQ.1) GO TO 575
      GO TO 520
C
C
C     ***************************************************************
C     *                     COMPUTE MODE SECTION                      *
C     *            OF THE 2173 COMPUTER SIMULATION PROGRAM            *
C     ***************************************************************
C
COMMENT      **COMPUTE MODE
 1000 IF(SIGNAL.EQ.1) WRITE(6,4030)
      IF(CHAR.EQ.1) GO TO 1060
 1005 DO 1010 I1=1,5
      CP(I1)=I(I1)
 1010 C(I1)=I(I1+7)
      DO 1015 I1=1,4
 1015 D(I1)=I(I1+23)
      DO 1020 I1=1,3
      SP(I1)=0
      IF(I(I1+15).EQ.ONE) SP(I1)=1
 1020 CONTINUE
      CODE=8*D(4)+4*D(3)+2*D(2)+D(1)+1
      IF(SIGNAL.EQ.1.OR.REGISTR.EQ.1) WRITE(6,4141)
      LIST1=LIST1+1
      IF(LIST1.GE.LIST) GO TO 415
C
COMMENT      **COUNTER SEARCH
      NCNT=16*C(5)+8*C(4)+4*C(3)+2*C(2)+C(1)+1
      NSECT=64*I(7)+32*I(6)+16*I(5)+8*I(4)+4*I(3)+2*I(2)+I(1)+1
      IF(CODE.EQ.1.OR.CODE.EQ.3.OR.CODE.EQ.7.OR.CODE.EQ.9.OR.CODE.EQ.9.
     1   OR.CODE.EQ.11.OR.CODE.EQ.12) GO TO 1025
C
```

```
COMMENT       ** NUMBER READ
      CHAN=NCHAN
      SECT=NSECT
      CALL UNLOAD
      IF(MFLAG.EQ.1) GO TO 15
C
COMMENT       **EXECUTE
 1025 PHASE=FSECT+MTIME
      IF(CODE.NE.1.AND.CODE.NE.9) GO TO 1030
      FSECT=ISECT+1
      GO TO 1035
 1030 IF(CODE.NE.7) GO TO 1031
      IF(A(24).EQ.ZERO) GO TO 1045
 1031 FSECT=NSECT
      GO TO 1040
 1035 IF(PHASE.LE.FSECT) GO TO 1040
      FSECT=PHASE
 1040 IF(FSECT.GT.128) FSECT=MOD(FSECT,128)
      IF(I(23).EQ.ZERO) GO TO 1045
      IF(SIGNAL.EQ.1) WRITE(6,4640)
      SECT=FSECT
      CALL FLASSTO
      IF(MFLAG.EQ.1) GO TO 15
C
 1045 GO TO (1255,1450,1233,1735,1715,1640,1723,1730,1850,1125,1115,1735
     1 ,1125,1145,1130,1165) CODE
C
 1050 GO TO (1245,1290,1095,1095,1245,1295,1095,1595,1245,1295,1595,1295
     1 ,1355,1420,1425,1605,1595,1230,1325,1280,1335,1310,1335,1210,1445
     2 ,1440,1595,1195,1430,1435,1430,1430) KCHAN
C
 1055 SECT=16*CB(5)+8*CB(4)+4*CB(3)+2*CB(2)+CB(1)
      IF(KCHAN.GE.17) GO TO 1490
      NUM=9
      GO TO (1540,1514,1573,1505,1564,1534,1594,1424,1530,1515,1509,1610
     1 ,1565,1535,1595,1625) KCHAN
C
COMMENT       **INSTRUCTION SEARCH
 1060 ICHAN=16*CP(5)+8*CP(4)+4*CP(3)+2*CP(2)+CP(1)+1
      IF(TRAS.EQ.1) GO TO 1072
      IF(I(23).EQ.ZERO) GO TO 1065
      CODE=8*I(16)+4*I(15)+2*I(14)+I(13)+1
      I1=MOD(FSECT-1,16)+1
      IF(CODE.EQ.I1) ISECT=FSECT
      IF(CODE.GT.I1) ISECT=FSECT+CODE-I1
      IF(CODE.LT.I1) ISECT=FSECT+16+CODE-I1
      IF(ISECT.GT.128) ISECT=MOD(ISECT,128)
      GO TO 1075
 1065 ISECT=64*I(19)+32*I(18)+16*I(17)+8*I(16)+4*I(15)+2*I(14)+I(13)+1
      GO TO 1075
 1072 ISECT=NSECT
      TRAS=NMOR=0
C
COMMENT       **INSTRUCTION READ
 1075 CHAN=ICHAN
      SECT=ISECT
      IREG=1
      CALL UNLOAD
      IF(MFLAG.EQ.1) GO TO 15
      IREG=5
      DO 1080 I1=1,24
 1080 I(I1)=N(I1)
      IF(REGIST.EQ.3) GO TO 1085
      REGIST=NREG(3)
      CALL REG2
C
 1085 IF(K.NE.SINGLE) GO TO 1035
      K=HALT
      GO TO 795
C
 1095 WRITE(6,4150)
      MTIME=1
      GO TO 1060
 1105 IF(REGIST.EQ.3) GO TO 1160
      REGIST=NREG(1)
      CALL REG2
      GO TO 1060
 1110 IF(REGIST.EQ.3) GO TO 1160
      A(12)=A(13)=ZERO
      REGIST=NREG(1)
      CALL REG2
      A(12)=A(13)=ZERO
      GO TO 1359
C
COMMENT     **TRANSFER   (TRA)
 1115 IF(SIGNAL.EQ.1) WRITE(6,4599)
      NTIME=FSECT=PHASE=6
      IPAS=1
      IF(CHAN.NE.27.AND.CHAN.NE.29.AND.CHAN.NE.79) GO TO 1120
      WRITE(5,4559)
      WRITE(6,4359)
      GO TO 15
 1120 DO 1125 I1=1,5
 1125 CF(I1)=C(I1)
      GO TO 1360
C
COMMENT     **TRANSFER ON MINUS   (TMI)
 1130 IF(SIGNAL.EQ.1) WRITE(6,4581)
      IF(A(24).EQ.1) GO TO 1115
      NTIME=1
      GO TO 1360
C
COMMENT     **CLEAR & ADD TO ACCUMULATOR   (CLA)
 1135 IF(SIGNAL.EQ.1) WRITE(6,4592)
      DO 1140 I1=1,24
 1140 A(I1)=M(I1)
      NTIME=1
      GO TO 1125
C
COMMENT     **ADD   (ADD)
 1145 IF(SIGNAL.EQ.1) WRITE(6,4583)
      NUM=24
 1150 I2=1
      NTIME=1
 1155 AK=ZERO
      DO 1160 I1=I2,NUM
      CODE=4*A(I1)+2*M(I1)+AK
      IF(M(I1).EQ.ONE.AND.A(I1).EQ.ONE) AK=ONE
      IF(M(I1).EQ.ZERO.AND.A(I1).EQ.ZERO) AK=ZERO
      A(I1)=ZERO
      IF(CODE.EQ.1.OR.CODE.EQ.2.OR.CODE.EQ.4.OR.CODE.EQ.7) A(I1)=ONE
 1160 CONTINUE
      IF(I2.EQ.14) GO TO 1110
      IF(NUM.EQ.24) GO TO 1125
      I2=14
      NUM=24
      GO TO 1155
C
COMMENT     **SUBTRACT   (SUB)
 1165 IF(SIGNAL.EQ.1) WRITE(6,4584)
      NUM=24
 1170 I2=1
      NTIME=1
 1175 AK=ZERO
      DO 1180 I1=I2,NUM
      CODE=4*A(I1)+2*M(I1)+AK
      IF(M(I1).EQ.ONE.AND.A(I1).EQ.ZERO) AK=ONE
      IF(M(I1).EQ.ZERO.AND.A(I1).EQ.ONE) AK=ZERO
      A(I1)=ZERO
      IF(CODE.EQ.1.OR.CODE.EQ.2.OR.CODE.EQ.4.OR.CODE.EQ.7) A(I1)=ONE
 1180 CONTINUE
      IF(I2.EQ.14) GO TO 1110
      IF(NUM.EQ.24) GO TO 1125
      I2=14
      NUM=24
      GO TO 1175
C
COMMENT     **SPLIT ADD   (SAD)
 1185 IF(SIGNAL.EQ.1) WRITE(6,4585)
      NUM=11
      GO TO 1150
C
COMMENT     **SPLIT SUBTRACT   (SSU)
 1190 IF(SIGNAL.EQ.1) WRITE(6,4586)
      NUM=11
      GO TO 1170
C
COMMENT     **HALT AND PROCEED   (HPR)
 1195 IF(SIGNAL.EQ.1) WRITE(6,4587)
C     K=J=PC=FC=1
      ISIG=STOPR=1
      GO TO 795
C
COMMENT     **COMPLEMENT   (COM)
 1200 IF(SIGNAL.EQ.1) WRITE(6,4588)
```

```
      NTIME=1
      DO 1213 I1=1,24
      IF(A(I1).EQ.ZERO) GO TO 1205
      A(I1)=ZERO
      GO TO 1210
 1205 A(I1)=ONE
 1210 CONTINUE
      AN=ONE
      DO 1215 I1=1,24
      IF(A(I1)+AN.NE.2) GO TO 1220
 1215 A(I1)=ZERO
 1220 A(I1)=ONE
      GO TO 1105
C
COMMENT      **MINUS MAGNITUDE    (MIM)
 1225 IF(SIGNAL.EQ.1) WRITE(6,4009)
      IF(A(24).EQ.0) GO TO 1200
      NTIME=1
      GO TO 1135
C
COMMENT      **LOGICAL AND TO ACCUMULATOR    (ANA)
 1230 IF(SIGNAL.EQ.1) WRITE(6,4010)
      NTIME=1
      DO 1235 I1=1,24
 1235 A(I1)=AND(A(I1),L(I1))
      GO TO 1135
C
COMMENT      **RESET DETECTOR    (RSD)
 1240 IF(SIGNAL.EQ.1) WRITE(6,4011)
      NTIME=1
      DR=0
      GO TO 1360
C
COMMENT      **BINARY OUTPUT A    (BOA)
 1245 IF(SIGNAL.EQ.1) WRITE(6,4012)
      NUM=1
 1250 NTIME=1
      IF(G(NUM).EQ.+1) GO TO 1265
      DO 1255 I1=17,24
      IF(A(I1)+ONE.EQ.ONE) GO TO 1260
 1255 A(I1)=ZERO
 1260 A(I1)=ONE
      GO TO 1280
 1265 I1=17
      IF(A(I1)-ONE.EQ.ZERO) GO TO 1275
      A(I1)=ONE
 1270 I1=I1+1
      IF(I1.EQ.25) GO TO 1280
      IF(A(I1).EQ.ONE) GO TO 1275
      A(I1)=ONE
      GO TO 1270
 1275 A(I1)=ZERO
 1280 G(NUM)=+1
      IF(A(24).EQ.ONE) G(NUM)=-1
      IF(G(NUM).EQ.-1) WRITE(6,4543) NUM
      IF(G(NUM).EQ.+1) WRITE(6,4552) NUM
      GO TO 1135
C
COMMENT      **BINARY OUTPUT B    (BOB)
 1285 IF(SIGNAL.EQ.1) WRITE(6,4013)
      NUM=2
      GO TO 1250
C
COMMENT      **BINARY OUTPUT C    (BOC)
 1290 IF(SIGNAL.EQ.1) WRITE(6,4014)
      NUM=3
      GO TO 1250
C
COMMENT      **DISCRETE OUTPUT    (DOA)
 1295 IF(SIGNAL.EQ.1) WRITE(6,4015)
      NTIME=1
      IF(DR.EQ.OFF) GO TO 1355
      DO 1300 I1=1,5
 1300 D(I1)=CR(I1)
      CODE=16*D(5)+8*D(4)+4*D(3)+2*D(2)+D(1)
      IF(CODE.EQ.0) GO TO 1363
      IF(CODE.EQ.5.OR.CODE.EQ.6.OR.CODE.EQ.7) GO TO 1362
      IF(CODE.EQ.11.AND.DR.EQ.1.OR.CODE.EQ.21.AND.DR.EQ.1) GO TO 1362
      WRITE(6,4542) CODE
      GO TO 1363
 1362 CODE=6
      WRITE(6,4543) CODE
      CODE=2*D(2)+D(1)
      WRITE(6,4243) CODE
      GO TO 1390
 1305 WRITE(6,4244)
      GO TO 1390
C
COMMENT      **DISCRETE INPUT 1   (DI1)
 1310 IF(SIGNAL.EQ.1) WRITE(6,4216)
      NTIME=1
      X1=X1+1
      IF(X1.GT.X2) GO TO 1315
      GO TO 1320
 1315 WRITE(6,4245)
      X1=X1-1
 1320 DO 1325 I1=1,19
 1325 A(I1)=X(I1,X1)
      DO 1330 I1=1,5
 1330 A(I1+19)=ZERO
      IF(PP.EQ.1) A(20)=ONE
      IF(FC.EQ.1) A(21)=ONE
      IF(P(3).EQ.1.AND.FC.EQ.0) A(22)=ONE
      IF(P(1).EQ.1) A(23)=ONE
      IF(P(2).EQ.1) A(24)=ONE
      GO TO 1135
C
COMMENT      **DISCRETE INPUT 2   (DI2)
 1335 IF(SIGNAL.EQ.1) WRITE(6,4217)
      NTIME=1
      Y1=Y1+1
      IF(Y1.GT.Y2) GO TO 1340
      GO TO 1345
 1340 WRITE(6,4246)
      Y1=Y1-1
 1345 DO 1350 I1=1,24
 1350 A(I1)=Y(I1,Y1)
      GO TO 1135
C
COMMENT      **VOLTAGE OUTPUT A   (VOA)
 1355 IF(SIGNAL.EQ.1) WRITE(6,4218)
      NUM=1
 1360 NTIME=1
      I2=0
      IF(I(4).EQ.1) I2=16
      DO 1365 I1=1,8
 1365 VO(I1)=A(I1+I2)
      VOLTAGE=0.
      DO 1375 I1=1,8
      IF(VO(I1).EQ.ONE) GO TO 1370
      VOLTAGE=VOLTAGE-VOLTS(I1)
      GO TO 1375
 1370 VOLTAGE=VOLTAGE+VOLTS(I1)
 1375 CONTINUE
      GO TO (1380,1385,1390) NUM
 1380 WRITE(6,4251)(VO(9-I1),I1=1,8),VOLTAGE
      GO TO 1395
 1385 WRITE(6,4252)(VO(9-I1),I1=1,8),VOLTAGE
      GO TO 1395
 1390 WRITE(6,4253)(VO(9-I1),I1=1,8),VOLTAGE
 1395 PHASE=4*P(3)+2*P(2)+P(1)
      IF(PHASE.NE.0) GO TO (1400,1405,1410,1405,1425,1415,1410) PHASE
      WRITE(6,4254)
      GO TO 1360
 1400 WRITE(6,4255) NUM
      GO TO 1360
 1405 WRITE(6,4256) NUM
      GO TO 1360
 1415 WRITE(6,4257) NUM
      GO TO 1360
 1415 WRITE(6,4258) NUM
      GO TO 1360
C
COMMENT      **VOLTAGE OUTPUT B   (VOB)
 1420 IF(SIGNAL.EQ.1) WRITE(6,4219)
      NUM=2
      GO TO 1360
C
COMMENT      **VOLTAGE OUTPUT C   (VOC)
 1425 IF(SIGNAL.EQ.1) WRITE(6,4220)
      NUM=3
      GO TO 1360
C
COMMENT      **LOAD PHASE REGISTER   (LPR)
 1430 IF(SIGNAL.EQ.1) WRITE(6,4221)
```

```
      NTIME=1
      DO 1435 I1=1,3
 1435 P(I1)=0
      IF(C(1).EQ.ONE) P(1)=1
      IF(C(2).EQ.ONE) P(2)=1
      IF(C(3).EQ.ONE) P(3)=1
      WRITE(6,4242)(P(4-I1),I1=1,3)
      GO TO 1560
C
COMMENT     **ENTER FIRE COMMAND   (EFC)
 1440 IF(SIGNAL.EQ.1) WRITE(6,4022)
      NTIME=1
      FC=1
      GO TO 1560
C
COMMENT     **HALT FIRE COMMAND   (HFC)
 1445 IF(SIGNAL.EQ.1) WRITE(6,4023)
      NTIME=1
      FC=0
      GO TO 1560
C
COMMENT     **SPLIT COMPARE & LIMIT   (SCL)
 1450 IF(SIGNAL.EQ.1) WRITE(6,4024)
      NTIME=2
      I2=0
 1455 CODE=0
      PHASE=0
      DO 1460 I1=1,10
 1460 CODE=CODE+2**(I1-1)*A(I2+I1)
      IF(A(I2+11).EQ.ZERO) GO TO 1496
      DO 1470 I1=1,11
      IF(K(I1+I2).EQ.ZERO) GO TO 1465
      A(I1+I2)=ZERO
      GO TO 1470
 1465 A(I1+I2)=ONE
 1470 CONTINUE
      DO 1475 I1=1,10
 1475 PHASE=PHASE+2**(I1-1)*A(I2+I1)
      IF(PHASE.LT.CODE) GO TO 1495
 1476 DO 1480 I1=1,11
 1480 A(I1+I2)=K(I1+I2)
 1495 IF(I2.EQ.13) GO TO 1516
      I2=13
      GO TO 1455
 1496 DO 1497 I1=1,10
 1497 PHASE=PHASE+2**(I1-1)*A(I2+I1)
      IF(PHASE.LE.CODE) GO TO 1476
      GO TO 1495
C
COMMENT     **CHARACTER OUTPUT   (COA)
 1500 IF(SIGNAL.EQ.1) WRITE(6,4025)
      IF(SECT.EQ.0) GO TO 1195
      NTIME=SECT+1
      CODE=8*A(24)+4*A(23)+2*A(22)+A(21)
      IF(CODE.LE.7) GO TO 1505
      DO 1503 I1=1,5
      IF(CODE.EQ.I1+7) GO TO 1504
 1503 CONTINUE
 1504 WRITE(6,4246) (A(25-I2),I2=1,4),MCODE(I1)
      GO TO 1510
 1505 WRITE(6,4247) (A(25-I1),I1=1,4), CODE
 1510 SECT=4
      GO TO 1520
C
COMMENT     **ACCUMULATOR LEFT SHIFT   (ALS)
 1514 NUM=1
 1515 IF(SIGNAL.EQ.1) WRITE(6,4026)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1185
 1520 DO 1530 I1=1,SECT
      DO 1525 I2=1,23
 1525 A(25-I2)=A(24-I2)
 1530 A(1)=ZERO
      GO TO 1195
C
COMMENT     **ACCUMULATOR RIGHT SHIFT   (ARS)
 1534 NUM=1
 1535 IF(SIGNAL.EQ.1) WRITE(6,4027)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1105
      I3=A(24)
      DO 1545 I1=1,SECT
      DO 1540 I2=1,23
 1540 A(I2)=A(I2+1)
 1545 A(24)=I3
      GO TO 1135
C
COMMENT      **SPLIT ACCUMULATOR LEFT SHIFT    (SAL)
 1549 NUM=1
 1550 IF(SIGNAL.EQ.1) WRITE(6,4028)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      DO 1560 I1=1,SECT
      DO 1555 I2=1,10
      A(25-I2)=A(24-I2)
 1555 A(12-I2)=A(11-I2)
 1560 A(14)=A(1)=ZERO
      GO TO 1110
C
COMMENT      **SPLIT ACCUMULATOR RIGHT SHIFT    (SAR)
 1564 NUM=1
 1565 IF(SIGNAL.EQ.1) WRITE(6,4029)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      I3=A(24)
      I4=A(11)
      DO 1575 I1=1,SECT
      DO 1570 I2=1,10
      A(I2)=A(I2+1)
 1570 A(13+I2)=A(14+I2)
      A(11)=I4
 1575 A(24)=I3
      GO TO 1110
C
COMMENT      **SPLIT LEFT WORD LEFT SHIFT    (SLL)
 1579 NUM=1
 1580 IF(SIGNAL.EQ.1) WRITE(6,4030)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      DO 1590 I1=1,SECT
      DO 1585 I2=1,10
 1585 A(25-I2)=A(24-I2)
 1590 A(14)=ZERO
      GO TO 1110
C
COMMENT      **SPLIT LEFT WORD RIGHT SHIFT    (SLR)
 1594 NUM=1
 1595 IF(SIGNAL.EQ.1) WRITE(6,4032)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      I3=A(24)
      DO 1605 I1=1,SECT
      DO 1600 I2=1,10
 1600 A(13+I2)=A(14+I2)
 1605 A(24)=I3
      GO TO 1110
C
COMMENT      **SPLIT RIGHT WORD LEFT SHIFT    (SRL)
 1609 NUM=1
 1610 IF(SIGNAL.EQ.1) WRITE(6,4031)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      DO 1620 I1=1,SECT
      DO 1615 I2=1,10
 1615 A(12-I2)=A(11-I2)
 1620 A(1)=ZERO
      GO TO 1110
C
COMMENT      **SPLIT RIGHT WORD RIGHT SHIFT    (SRR)
 1624 NUM=1
 1625 IF(SIGNAL.EQ.1) WRITE(6,4033)
```

```
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1118
      I4=A(11)
      DO 1635 I1=1,SECT
      DO 1636 I2=1,10
 1636 A(I2)=A(I2+1)
 1635 A(11)=I4
      GO TO 1119
C
COMMENT      **MULTIPLY    (MPY)
 1540 IF(SIGNAL.EQ.1) WRITE(6,4034)
      NTIME=13
      I2=1
      I3=I6=23
      NUM=-23
      DO 1650 I1=1,24
 1650 L(I1)=A(I1)
      IF(REGIST2.EQ.0  GO TO 1655
      REGIST=MREG(2)
      CALL REG2
 1655 COMREG(1)=COMREG(2)=ZERO
      I5=I3+1
      IF(A(I5).EQ.ZERO) GO TO 1670
      DO 1665 I1=I2,I3
      IF(A(I1).EQ.ONE) GO TO 1660
      A(I1)=ONE
      GO TO 1665
 1660 A(I1)=ZERO
 1665 CONTINUE
 1670 IF(M(I5).EQ.ZERO) GO TO 1685
      DO 1680 I1=I2,I3
      IF(M(I1).EQ.ONE) GO TO 1675
      M(I1)=ONE
      GO TO 1680
 1675 M(I1)=ZERO
 1680 CONTINUE
 1685 DO 1690 I1=1,I5
      COMREG(1)=SHIFT(COMREG(1),1)
      COMREG(2)=SHIFT(COMREG(2),1)
      COMREG(1)=OR(COMREG(1),A(I5-I1))
 1690 COMREG(2)=OR(COMREG(2),M(I5-I1))
      IF(A(I5).EQ.ONE) COMREG(1)=-(COMREG(1)+ONE)
      IF(M(I5).EQ.ONE) COMREG(2)=-(COMREG(2)+ONE)
      I4=COMREG(1)*COMREG(2)
      I4=SHIFT(I4,NUM)
      IF(A(I5).EQ.ONE.AND.M(I5).EQ.ONE) GO TO 1691
      IF(A(I5).EQ.ZERO.AND.M(I5).EQ.ONE) GO TO 1692
      GO TO 1693
 1691 A(I5)=ZERO
      GO TO 1694
 1692 A(I5)=ONE
 1693 IF(A(I5).EQ.ONE) I4=I4+ONE
 1694 DO 1695 I1=I2,I3
      A(I1)=AND(I4,ONE)
 1695 I4=SHIFT(I4,-1)
      IF(I2.EQ.14) GO TO 1118
      IF(I3.EQ.23) GO TO 1175
      I2=14
      I3=23
      GO TO 1555
C
COMMENT      **MULTIPLY MODIFIED    (MPM)
 1700 IF(SIGNAL.EQ.1) WRITE(6,4035)
      DO 1705 I1=1,2
      IF(C(I1).EQ.ZERO.AND.P(I1).EQ.1) GO TO 1704
      IF(C(I1).EQ.ONE.AND.P(I1).EQ.1) C(I1)=ZERO
      GO TO 1705
 1704 C(I1)=ONE
 1705 CONTINUE
      IF(FC.EQ.1) GO TO 1710
      IF(C(3).EQ.ZERO.AND.P(3).EQ.1) GO TO 1709
      IF(C(3).EQ.ONE.AND.P(3).EQ.1) C(3)=ZERO
      GO TO 1710
 1709 C(3)=ONE
 1710 NSECT=16*C(5)+8*C(4)+4*C(3)+2*C(2)+C(1)+1
      SECT=NSECT
      CALL UNLOAD
      IF(MFLAG.EQ.1) GO TO 15
      GO TO 1540
C
```

```
COMMENT       **SPLIT MULTIPLY   (SMP)
 1715 IF(SIGNAL.EQ.1) WRITE(6,4935)
      NTIME=7
      I2=1
      I3=I6=19
      NUM=-13
      DO 1716 I1=1,11
      L(I1)=A(I1+13)
 1716 L(I1+13)=A(I1)
      L(1)=L(13)=ZERO
      IF(REGISTR.EQ.3) GO TO 1655
      REGIST=NREG(2)
      CALL REG2
      GO TO 1555
C
COMMENT       **SPLIT MULTIPLY MODIFIED   (SMM)
 1720 IF(SIGNAL.EQ.1) WRITE(6,4937)
      DO 1725 I1=1,2
      IF(C(I1).EQ.ZERO.AND.P(I1).EQ.1) GO TO 1724
      IF(C(I1).EQ.ONE.AND.P(I1).EQ.1) C(I1)=ZERO
      GO TO 1725
 1724 C(I1)=ONE
 1725 CONTINUE
      IF(FC.EQ.1) GO TO 1730
      IF(C(3).EQ.ZERO.AND.P(3).EQ.1) GO TO 1729
      IF(C(3).EQ.ONE.AND.P(3).EQ.1) C(3)=ZERO
      GO TO 1730
 1729 C(3)=ONE
 1730 CHAN=16*C(5)+8*C(4)+4*C(3)+2*C(2)+C(1)+1
      SECT=NSECT
      CALL UNLOAD
      IF(NFLAG.EQ.1) GO TO 15
      GO TO 1715
C
COMMENT       **STORE ACCUMULATOR   (STO)
 1735 IF(SIGNAL.EQ.1) WRITE(6,4938)
      NTIME=1
      CHAN=NCHAN
      SECT=NSECT
      IF(CHAN.NE.22.AND.CHAN.NE.23.AND.CHAN.NE.24.AND.CHAN.NE.25.AND.
     1   CHAN.NE.26) SECT=SECT-2
      IF(SECT.EQ.-1) SECT=127
      IF(SECT.EQ.0) SECT=128
      CALL STORE
      IF(NFLAG.EQ.1) GO TO 15
      IF(CODE.EQ.4) GO TO 1135
      GO TO 1363
C
C
C     ************************************************************
C     *                      FORMAT SECTION                      *
C     *          OF THE 2173 COMPUTER SIMULATION PROGRAM         *
C     ************************************************************
C
COMMENT       **MESSAGES & TRANSLATION FORMAT STATEMENTS
 2300 FORMAT(/,3X,56("*"),/,3X,"**",52X,"**",/,3X,"**",13X,"2173 COMPUT"
     1  "ER",23X,"**",/,3X,"**",17X,"SIMULATION PROGRAM",17X,"**",/,3X,
     2  "**",52X,"**",/,3X,"**",4X,"DATE= ",A9,14X,"TIME= ",A9,4X,"**",
     3  /,3X,"**",52X,"**",/,3X,56("*"),/)
 2301 FORMAT(/,33X,56("*"),/,33X,"**",52X,"**",/,33X,"**",19X,"2173 COM"
     1  "PUTER",23X,"**"/,33X,"**",17X,"SIMULATION PROGRAM",17X,"**",/,
     2  33X,"**",52X,"**",/,33X,"**",4X,"DATE= ",A9,14X,"TIME= ",A9,4X,
     3  "**",/,33X,"**",11X,3A10,11X,"**",/,33X,"**",52X,"**",/,33X,56
     4  ("*"),/)
 2305 FORMAT(" TO RUN ANOTHER PROGRAM TYPE "RUN": TO STOP TYPE "HALT" -
     1")
 2310 FORMAT(A3)
 2315 FORMAT( / ,23X,"**  PRINTOUT OF INPUT PROGRAM  **"/" (ENTER PROGRA
     1M)"/)
 2320 FORMAT(7A10,A2)
 2321 FORMAT(I4 ,7A10,A2)
 2322 FORMAT(A1.3A10)
 2323 FORMAT(" IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOU
     1R NAME": OTHER-WISE TYPE "N" - ")
 2325 FORMAT(/ /,21X,"**  RESULTS OF SIMULATION  **"/)
 2335 FORMAT(72A1)
 2375 FORMAT(/," CONTINUE PROGRAM",/)
 2340 FORMAT(" THE FOLLOWING DATA IS NOT ALLOWED:  ",72A1)
 2345 FORMAT(" LOAD CODES MUST BE IN BINARY WHEN BINARY IS SPECIFIED")
 2350 FORMAT(" THE FOLLOWING INPUT CODE IS INVALID:  ",72A1)
 2355 FORMAT(" COMPUTER IS NOT IN WAIT MODE - DATA CANNOT BE ENTERED - P
     1ROGRAM TERMINATED")
 2060 FORMAT(' AN FS(ON) SIGNAL MUST FOLLOW AN IN(ON) SIGNAL TO PUT MACH SIM     1404
     1INE IN WAIT    MODE - DATA IGNORED')                                SIM     1405
 2065 FORMAT(' COMPUTE SWITCH MODE SPECIFIED INCORRECTLY')                SIM     1406
 2070 FORMAT(' SIGNAL ON - MODES WILL BE TRACED',/)                       SIM     1407
 2075 FORMAT(' SIGNAL OFF - MODES WILL NOT BE TRACED FURTHER',/)          SIM     1408
 2080 FORMAT(/,' INPUT SOURCE - MAG TAPE',/)                              SIM     1409
 2085 FORMAT(' THE FOLLOWING INPUT DATA ON MAG TAPE IS INVALID - ',A1)    SIM     1410
 2090 FORMAT(' EXECUTE SEGMENT SPECIFIED INCORRECTLY - DEFAULT VALUE OF SIM     1411
     1 50 ASSUMED')                                                       SIM     1412
 2095 FORMAT(' NO. OF EXECUTIONS SPECIFIED = ',I4)                        SIM     1413
 2100 FORMAT(' COLD-STORAGE WRITE SWITCH SPECIFIED INCORRECTLY')          SIM     1414
 2105 FORMAT(' DISCRETE SWITCH SPECIFIED INCORRECTLY')                    SIM     1415
 2110 FORMAT(' POWER-ON/OFF DATA IS INCORRECT')                           SIM     1416
 2115 FORMAT(/,' POWER HAS BEEN TURNED OFF')                              SIM     1417
 2120 FORMAT(' POWER HAS BEEN TURNED ON',/)                               SIM     1418
 2125 FORMAT(' CHANNEL SPECIFIED CANNOT BE LOADED - PROGRAM TERMINATED') SIM     1419
 2130 FORMAT(' IN(ON) CANNOT BE SPECIFIED WITH FS(OFF) - PROGRAM TERMINA SIM     1420
     1TED')                                                               SIM     1421
 2135 FORMAT(' NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM T SIM     1422
     1ERMINATED')                                                         SIM     1423
 2140 FORMAT(///5X,'** END OF PROGRAM',5X,'EXECUTION TIME= ',F8.3,2X,     SIM     1424
     1 'SEC')                                                             SIM     1425
 2145 FORMAT(' <HALT> MUST BE SPECIFIED BEFORE <RUN> AFTER AN HF* INST SIM     1426
     1RUCTION -      PROGRAM TERMINATED')                                 SIM     1427
 2150 FORMAT(' MEMORY HAS BEEN INITIALIZED')                              SIM     1428
C                                                                         SIM     1429
COMMENT     **NONCOMPUTE MODE FORMAT STATEMENTS                           SIM     1430
 3000 FORMAT(' MASTER RESET SEQUENCE')                                    SIM     1431
 3005 FORMAT(' PREPARE TO OPERATE MODE')                                  SIM     1432
 3010 FORMAT(' SYNC BIT COUNTER 1 MODE')                                  SIM     1433
 3015 FORMAT(' SYNC BIT COUNTER 2 MODE')                                  SIM     1434
 3020 FORMAT(/,' C(4)'C(2)C(1)' IDLE SUB-MODE OF MANUAL HALT')            SIM     1435
 3025 FORMAT(' D(4)'C(2)C(1)' IDLE SUB-MODE OF MANUAL HALT')              SIM     1436
 3030 FORMAT(' INTERLOCK SUB-MODE OF MANUAL HALT')                        SIM     1437
 3035 FORMAT(' PREPARE TO LOAD SUB-MODE OF MANUAL HALT')                  SIM     1438
 3040 FORMAT(' CIRCULATES BETWEEN PREPARE TO LOAD AND INTERLOCK SUB-MODE SIM     1439
     1S OF MANUAL    HALT')                                               SIM     1440
 3045 FORMAT(' D(4)'C(2)C(1) IDLE SUB-MODE OF MANUAL HALT')               SIM     1441
 3050 FORMAT(' CIRCULATES BETWEEN D(4)'C(2)C(1) IDLE, INTERLOCK, AND C(4 SIM     1442
     1)'C(2)C(1)'    IDLE SUB-MODES OF MANUAL HALT')                      SIM     1443
 3055 FORMAT(' PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT',/)             SIM     1444
 3060 FORMAT(' WAIT MODE',/)                                              SIM     1445
 3065 FORMAT(' VERIFY ERROR')                                             SIM     1446
 3070 FORMAT(' PREPARE TO SAMPLE MODE')                                   SIM     1447
 3075 FORMAT(' SAMPLE CODE MODE')                                         SIM     1448
 3080 FORMAT(' PARITY CHECK MODE')                                        SIM     1449
 3085 FORMAT(' PROCESS CODE - OCTAL')                                     SIM     1450
 3090 FORMAT(' PROCESS CODE - HALT')                                      SIM     1451
 3095 FORMAT(' PROCESS CODE - LOCATION')                                  SIM     1452
 3100 FORMAT(' PROCESS CODE - FILL')                                      SIM     1453
 3105 FORMAT(' PROCESS CODE - VERIFY')                                    SIM     1454
 3110 FORMAT(' PROCESS CODE - START TO COMPUTE')                          SIM     1455
 3115 FORMAT(' PROCESS CODE - ENTER')                                     SIM     1456
 3120 FORMAT(' ENTER - IDLE SUB-MODE OF FILL-VERIFY')                     SIM     1457
 3125 FORMAT(' ENTER - NUMBER SEARCH SUB-MODE OF FILL-VERIFY')            SIM     1458
 3130 FORMAT(' ENTER - WAIT 2 W.T. SUB-MODE OF FILL-VERIFY')              SIM     1459
 3135 FORMAT(' ENTER - EXECUTE SUB-MODE OF FILL-VERIFY')                  SIM     1460
 3140 FORMAT(' PROCESS CODE - CLEAR')                                     SIM     1461
 3145 FORMAT(' PROCESS CODE - DELETE')                                    SIM     1462
 3150 FORMAT(' PARITY ERROR')                                             SIM     1463
 3155 FORMAT(' PROGRAM HALT MODE')                                        SIM     1464
C                                                                         SIM     1465
COMMENT     **COMPUTE MODE FORMAT STATEMENTS                              SIM     1466
 4000 FORMAT(' TRANSFER INSTRUCTION - (TRA)')                             SIM     1467
 4001 FORMAT(' TRANSFER ON MINUS INSTRUCTION - (TMI)')                    SIM     1468
 4002 FORMAT(' CLEAR & ADD INSTRUCTION - (CLA)')                          SIM     1469
 4003 FORMAT(' ADD INSTRUCTION - (ADD)')                                  SIM     1470
 4004 FORMAT(' SUBTRACT INSTRUCTION - (SUB)')                             SIM     1471
 4005 FORMAT(' SPLIT ADD INSTRUCTION - (SAD)')                            SIM     1472
 4006 FORMAT(' SPLIT SUBTRACT INSTRUCTION - (SSU)')                       SIM     1473
 4007 FORMAT(' HALT AND PROCEED INSTRUCTION - (HPR)')                     SIM     1474
 4008 FORMAT(' COMPLEMENT INSTRUCTION - (COM)')                           SIM     1475
 4009 FORMAT(' SHIFT MAGNITUDE INSTRUCTION - (SHM)')                      SIM     1476
 4010 FORMAT(' LOGICAL AND TO ACCUMULATOR INSTRUCTION - (ANA)')           SIM     1477
 4011 FORMAT(' RESET DETECTOR INSTRUCTION - (RSD)')                       SIM     1478
 4012 FORMAT(' BINARY OUTPUT "A" INSTRUCTION - (BOA)')                    SIM     1479
 4013 FORMAT(' BINARY OUTPUT "B" INSTRUCTION - (BOB)')                    SIM     1480
 4014 FORMAT(' BINARY OUTPUT "C" INSTRUCTION - (BOC)')                    SIM     1481
 4015 FORMAT(' DISCRETE OUTPUT INSTRUCTION - (DOA)')                      SIM     1482
 4016 FORMAT(' DISCRETE INPUT "A" INSTRUCTION - (DIA)')                   SIM     1483
 4017 FORMAT(' DISCRETE INPUT "B" INSTRUCTION - (DIB)')                   SIM     1484
 4018 FORMAT(' VOLTAGE OUTPUT "A" INSTRUCTION - (VOA)')                   SIM     1485
 4019 FORMAT(' VOLTAGE OUTPUT "B" INSTRUCTION - (VOB)')                   SIM     1486
```

```
4320 FORMAT(* VOLTAGE OUTPUT "C" INSTRUCTION - (VOC)*)                   SIM     1487
4321 FORMAT(* LOAD PHASE REGISTER INSTRUCTION - (LPR)*)                   SIM     1488
4322 FORMAT(* ENTER FINE COUNTDOWN INSTRUCTION - (EFC)*)                  SIM     1489
4323 FORMAT(* HALT FINE COUNTDOWN INSTRUCTION - (HFC)*)                   SIM     1490
4324 FORMAT(* SPLIT COMPARE & LIMIT INSTRUCTION - (SCL)*)                 SIM     1491
4325 FORMAT(* CHARACTER OUTPUT INSTRUCTION - (COA)*)                      SIM     1492
4326 FORMAT(* ACCUMULATOR LEFT SHIFT INSTRUCTION - (ALS)*)                SIM     1493
4327 FORMAT(* ACCUMULATOR RIGHT SHIFT INSTRUCTION - (ARS)*)               SIM     1494
4328 FORMAT(* SPLIT ACCUMULATOR LEFT SHIFT INSTRUCTION - (SAL)*)          SIM     1495
4329 FORMAT(* SPLIT ACCUMULATOR RIGHT SHIFT INSTRUCTION - (SAR)*)         SIM     1496
4330 FORMAT(* SPLIT LEFT WORD LEFT SHIFT INSTRUCTION - (SLL)*)            SIM     1497
4331 FORMAT(* SPLIT RIGHT WORD LEFT SHIFT INSTRUCTION - (SRL)*)           SIM     1498
4332 FORMAT(* SPLIT LEFT WORD RIGHT SHIFT INSTRUCTION - (SLR)*)           SIM     1499
4333 FORMAT(* SPLIT RIGHT WORD RIGHT SHIFT INSTRUCTION - (SRR)*)          SIM     1500
4334 FORMAT(* MULTIPLY INSTRUCTION - (MPY)*)                              SIM     1501
4335 FORMAT(* MULTIPLY MODIFIED INSTRUCTION - (MPM)*)                     SIM     1502
4336 FORMAT(* SPLIT MULTIPLY INSTRUCTION - (SMP)*)                        SIM     1503
4337 FORMAT(* SPLIT MULTIPLY MODIFIED INSTRUCTION - (SMM)*)               SIM     1504
4338 FORMAT(* STORE ACCUMULATOR INSTRUCTION - (STO)*)                     SIM     1505
4339 FORMAT(/,* COMPUTE MODE*)                                            SIM     1506
4340 FORMAT(* FLAG STORE*)                                                SIM     1507
4341 FORMAT(1H )                                                          SIM     1508
4342 FORMAT(* PHASE REGISTER - P(3-1)= *,3I1)                             SIM     1509
4343 FORMAT(* DISCRETE OUTPUT LINE D*,I2,*C HAS A "1" OUTPUT SIGNAL*)     SIM     1510
4344 FORMAT(* DISCRETE SWITCH IS OFF - DISCRETE OUTPUTS ARE DISABLED*)    SIM     1511
4345 FORMAT(* X-DISCRETE INPUTS USED HAVE EXCEEDED THOSE SUPPLIED - LAS SIM     1512
    1T VALUES GIVEN WILL BE USED*)                                        SIM     1513
4346 FORMAT(* Y-DISCRETE INPUTS USED HAVE EXCEEDED THOSE SUPPLIED - LAS SIM     1514
    1T VALUES GIVEN WILL BE USED*)                                        SIM     1515
4347 FORMAT(* BINARY CHARACTER OUTPUT - *,4O1,3X,*HEXIDECIMAL CHARACTER SIM     1516
    1 OUTPUT - *,Z1)                                                      SIM     1517
4348 FORMAT(* BINARY CHARACTER OUTPUT - *,4O1,3X,*HEXIDECIMAL CHARACTER SIM     1518
    1 OUTPUT - *,Z1)                                                      SIM     1519
4349 FORMAT(* BINARY OUTPUT ON LINE B*,I1,*1 OF -1*)                     SIM     1520
4350 FORMAT(* BINARY OUTPUT ON LINE B*,I1,*0 OF +1*)                     SIM     1521
4351 FORMAT(* V1(3-1)= *,3I1,* WITH A VOLTAGE OUTPUT OF*,F7.2,* VOLTS*) SIM     1522
4352 FORMAT(* V2(3-1)= *,3I1,* WITH A VOLTAGE OUTPUT OF*,F7.2,* VOLTS*) SIM     1523
4353 FORMAT(* V3(3-1)= *,3I1,* WITH A VOLTAGE OUTPUT OF*,F7.2,* VOLTS*) SIM     1524
4354 FORMAT(* THE PHASE REGISTER IS IN THE IDLE CONFIGURATION: NO VOLTA SIM     1525
    1GE OUTPUT*)                                                          SIM     1526
4355 FORMAT(* VOLTAGE OUTPUT IS ON LINE VC*,I1,*0*)                      SIM     1527
4356 FORMAT(* VOLTAGE OUTPUT IS ON LINE VC*,I1,*1*)                      SIM     1528
4357 FORMAT(* VOLTAGE OUTPUT IS ON LINE VC*,I1,*2*)                      SIM     1529
4358 FORMAT(* VOLTAGE OUTPUT IS ON LINE VC*,I1,*3*)                      SIM     1530
4359 FORMAT(* A TRANSFER IS NOT ALLOWED TO L-REG, Y-, OR R- LOOPS - PRO SIM     1531
    1GRAM TERMINATED*)                                                    SIM     1532
4360 FORMAT(* THE X-INSTRUCTION REQUESTED IS NOT AN INSTRUCTION*)         SIM     1533
     END                                                                  SIM     1534
```

```
      BLOCK DATA                                                        SIM  1535
                                                                        SIM  1536
      INTEGER ON, ZERO, ONE                                             SIM  1537
                                                                        SIM  1538
      COMMON/ANDATA/ ON, ZERO, ONE, KREG(12), NN(27), NBLANK, NCOMMA,   SIM  1539
     1               NLPAREN, NRPAREN, NINES                            SIM  1540
                                                                        SIM  1541
      DATA KREG/1HA, 1HL, 1HI, 1HM, 1HU, 1HF, 1HE, 1HH, 1HV, 1HS/       SIM  1542
                                                                        SIM  1543
      DATA ZERO/33/, ONE/13/, NINES/999999999/                          SIM  1544
                                                                        SIM  1545
      DATA NN/1HO, 1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1HL, 1HF,    SIM  1546
     1        1HN, 1HC, 1HE, 1HD, 1HT, 1HK, 1HP, 1HI, 1HM, 1HO, 1HB,    SIM  1547
     2        1HS, 1HR, 1HX, 1HY, 1HW/                                  SIM  1548
                                                                        SIM  1549
      DATA NCOMMA/1H,/, NLPAREN/1H(/, NRPAREN/1H)/, NBLANK/1H /         SIM  1550
      DATA ON/2HON/                                                     SIM  1551
                                                                        SIM  1552
      END                                                               SIM  1553
      SUBROUTINE STORE
      INTEGER A(24), CHAN, CODE, COMREG(24,3), DD, E(8), EN, F(6), FC
      INTEGER H(15), DN, ONE, PHASE, R(24,4), REGIST, REGISTR, RI, RK
      INTEGER SR(3), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,19), XI
      INTEGER Y(24,13), YI, ZERO, DISPOSE
      COMMON A, CHAN, CODE, COMREG, DD, E, EN, F, FC, H, I(24), IREG
      COMMON L(24), M(128,21), N(24), NCOL, NFLAG, NUM, NWORD(72), PHASE
      COMMON SR, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK
      COMMON X, XI, Y, YI, DISPOSE
      COMMON/NUMDATA/ ON, ZERO, ONE, NREG(16), NN(27), NBLANK, NCOMMA,
     1                NLPAREN, NRPAREN, NIDES
      IF(CHAN.EQ.21) GO TO 5
      IF(CHAN.EQ.22.OR.CHAN.EQ.23.OR.CHAN.EQ.24) GO TO 1
      IF(CHAN.EQ.25.OR.CHAN.EQ.26.OR.CHAN.EQ.27.OR.CHAN.EQ.28) GO TO 10
      IF(CHAN.EQ.29) GO TO 15
      IF(CHAN.EQ.30) GO TO 35
      IF(EN.EQ.ON) GO TO 1
      WRITE(6,100)
      WRITE(4,100)
      NFLAG=1
      RETURN
    1 CALL LOAD
      RETURN
    5 IF(DD.EQ.ON) GO TO 1
      WRITE(6,110)
      WRITE(4,110)
      NFLAG=1
      RETURN
   10 WRITE(6,105)
      WRITE(4,105)
      NFLAG=1
      RETURN
   15 I2=MOD(SECT-1,4)+1
      IF(FC.EQ.1) GO TO 25
      VK=ZERO
      DO 20 I1=1,24
      CODE=4*A(I1)+2*V(I1,I2)+VK
      IF(V(I1,I2).EQ.ONE.AND.A(I1).EQ.ONE) VK=ONE
      IF(V(I1,I2).EQ.ZERO.AND.A(I1).EQ.ZERO) VK=ZERO
      V(I1,I2)=ZERO
      IF(CODE.EQ.1.OR.CODE.EQ.2.OR.CODE.EQ.4.OR.CODE.EQ.7) V(I1,I2)=ONE
   20 CONTINUE
   21 IF(REGISTR.EQ.0) RETURN
      REGIST=NREG(3)
      CALL REG2
      RETURN
   25 DO 30 I1=1,24
   30 V(I1,I2)=A(I1)
      GO TO 21
   35 IF(FC.EQ.1) GO TO 55
      I4=MOD(SECT-1, )+1
      I2=1
      I3=11
   40 RK=ZERO
      DO 45 I1=I2,I3
      CODE=4*A(I1)+2 R(I1,I4)+RK
      IF(R(I1,I4).EQ ONE.AND.A(I1).EQ.ONE) RK=ONE
      IF(R(I1,I4).EQ ZERO.AND.A(I1).EQ.ZERO) RK=ZERO
      R(I1,I4)=ZERO
      IF(CODE.EQ.1.O .CODE.EQ.2.OR.CODE.EQ.4.OR.CODE.EQ.7) R(I1,I4)=ONE
   45 CONTINUE
      IF(I3.EQ.24) GO TO 50
      I2=14
      I3=24
      GO TO 40
   50 IF(REGISTR.EQ.0) RETURN
      REGIST=NREG(15)
      CALL REG2
   55 RETURN
  100 FORMAT(* STORAGE CANNOT TAKE PLACE IN COLD STORAGE CHANNELS IF COL
     1D-STORAGE      WRITE SWITCH IS OFF - PROGRAM TERMINATED*)
  105 FORMAT(* STORAGE IS NOT ALLOWED IN SINGLE-LOOPS (R,I,L, OR U) - PR
     1OGRAM         TERMINATED*)
  110 FORMAT(* STORAGE CANNOT TAKE PLACE IN CHANNEL 50 IF DISCRETE SWITC
     1H IS OFF -     PROGRAM TERMINATED*)
      END
      SUBROUTINE LOAD
      INTEGER A(24), CHAN, CODE, COMREG(24,2), DD, E(8), EN, F(4), FC
      INTEGER H(16), CS, CNE, PHASE, R(24,4), REGIST, REGISTR, RI, RK
      INTEGER SB(2), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,12), XI
      INTEGER Y(24,15), YI, ZERO, DISPOSE
      COMMON A, CHAN, CODE, COMREG, DD, E, EN, F, FC, H, I(24), IREG
      COMMON L(24), M(128,21), N(24), ACCL, NFLAG, SUY, NAOFD(72), PHASE
      COMMON SB, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK
      COMMON X, XI, Y, YI, DISPOSE
      COMMON/NDATA/ ON, ZERO, ONE, NREG(18), NN(27), NBLANK, NCOMMA,
     1              NLPAREN, NRPAREN, NIKES
      COMREG(1)=ZERO
      DO 1 I1=1,24
      I2=SHIFT(COMREG(1),1)
    1 COMREG(1)=OR(I2,A(25-I1))
      IF(CHAN.EQ.21) GO TO 10
      IF(CHAN.EQ.22) GO TO 15
      IF(CHAN.EQ.23) GO TO 20
      IF(CHAN.EQ.24) GO TO 25
      IF(CHAN.EQ.25) GO TO 30
      IF(SB.EQ.ON) GO TO 5
      WRITE(5,100)
      WRITE(4,100)
      NFLAG=1
      RETURN
    5 M(SECT,CHAN)=COMREG(1)
      RETURN
   10 IF(DD.EQ.ON) GO TO 5
      WRITE(5,105)
      WRITE(4,105)
      NFLAG=1
      RETURN
   15 I2=MOD(SECT-1,4)+1
      F(I2)=COMREG(1)
      REGIST=NREG(6)
      GO TO 35
   20 I2=MOD(SECT-1,16)+1
      H(I2)=COMREG(1)
      REGIST=NREG(8)
      GO TO 35
   25 I2=MOD(SECT-1,8)+1
      E(I2)=COMREG(1)
      REGIST=NREG(7)
      GO TO 35
   30 U=COMREG(1)
      REGISTR=NREG(5)
   35 IF(REGISTR.EQ.0) RETURN
      CALL REG2
      RETURN
  100 FORMAT(* COLD STORAGE MEMORY CANNOT BE LOADED IF COLD-STORAGE WRIT
     1E SWITCH IS    OFF - PROGRAM TERMINATED*)
  105 FORMAT(* HOT STORAGE MEMORY CANNOT BE LOADED IF DISCRETE SWITCH IS
     1 OFF - PROGRAM TERMINATED*)
      END
      SUBROUTINE UNLOAD
      INTEGER A(24), CHAN, CODE, COMRES(24,2), DD, E(8), EN, F(4), FC
      INTEGER H(16), CN, CNE, PHASE, R(24,4), REGIST, REGISTR, RI, RK
      INTEGER S3(3), SECT, SIGNAL, U, V(24,4), VI, VK, X(10,10), XI
      INTEGER Y(24,10), YI, ZERO, DISPOSE
      COMMON A, CHAN, CODE, COMRES, DD, E, EN, F, FC, H, I(24), IREG
      COMMON L(24), M(126,21), N(24), NCOL, NFLAG, NUM, NWORD(72), PHASE
      COMMON S3, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK
      COMMON X, XI, Y, YI, DISPOSE
      COMMON/MYDATA/ ON, ZERO, ONE, NREG(10), NN(27), NBLANK, NCOMMA,
     1               NLPAREN, NRPAREN, NINES
      NUM=0
      DO 1 I1=1,11
      IF(CHAN.EQ.(21+I1)) GO TO (5,10,15,20,25,35,45,50,60,70,75) I1
    1 CONTINUE
      I2=M(SECT,CHAN)
      GO TO 80
    5 I2=MOD(SECT-1,4)+1
      I2=F(I2)
      GO TO 80
   10 I2=MOD(SECT-1,16)+1
      I2=H(I2)
      GO TO 80
   15 I2=MOD(SECT-1,8)+1
      I2=E(I2)
      GO TO 80
   20 I2=U
      GO TO 80
   25 DO 30 I1=1,24
   30 N(I1)=A(I1)
      GO TO 110
   35 DO 40 I1=1,24
   40 N(I1)=L(I1)
   45 GO TO 110
   50 IF(VK.EQ.1) NUM=1
      I2=MOD(SECT-1,4)+1
      DO 55 I1=1,24
   55 N(I1)=V(I1,I2)
      GO TO 90
   60 IF(RK.EQ.1) NUM=1
      I2=MOD(SECT-1,4)+1
      DO 65 I1=1,24
   65 N(I1)=R(I1,I2)
      GO TO 90
   70 I2=MOD(SECT-1,16)+1
      I2=MOD(I2+8,16)
      I2=H(I2)
      GO TO 80
   75 I2=MOD(SECT-1,8)+1
      I2=MOD(I2+4,8)
      I2=F(I2)
   80 IF(I2.EQ.NINES) GO TO 115
      DO 85 I1=1,24
      N(I1)=AND(I2,ONE)
   85 I2=SHIFT(I2,-1)
   90 IF(NUM.EQ.1) GO TO 95
      GO TO 110
   95 DO 105 I1=1,24
      IF(N(I1).EQ.ONE) GO TO 100
      N(I1)=ONE
      GO TO 105
  100 N(I1)=ZERO
  105 CONTINUE
  110 IF(IREG.EQ.1) RETURN
      IF(REGISTR.EQ.3) RETURN
      REGIST=NREG(4)
      CALL RES2
      RETURN
  115 IF(IREG.EQ.1) GO TO 125
      WRITE(6,130)
      WRITE(4,130)
  120 NFLAG=1
      RETURN
  125 WRITE(6,135)
      WRITE(4,135)
      GO TO 120
  130 FORMAT(" OPERAND ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED")
  135 FORMAT(" INSTRUCTION ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED"
     1)
      END
      SUBROUTINE FLAGSTO                                                SIM  1764
      INTEGER A(24), CHAN, CODE, COMRES(24,2), DO, E(3), EN, F(4), FC   SIM  1765
      INTEGER M(15), OK, ONE, PHASE, R(24,4), REGIST, REGISTR, RI, RK   SIM  1766
      INTEGER SB(2), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,12), XI     SIM  1767
      INTEGER Y(24,13), YI, ZERO, DISPOSE                               SIM  1768
      COMMON A, CHAN, CODE, COMRES, DO, E, EN, F, FC, M, I(24), IREG    SIM  1769
      COMMON L(24), M(128,21), K(24), KCOL, KFLAG, NUM, NWORD(72), PHASE SIM  1770
      COMMON SB, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK SIM  1771
      COMMON X, XI, Y, YI, DISPOSE                                      SIM  1772
      COMMON/MYDATA/ ON, ZERO, ONE, NREG(12), NA(27), NBLANK, NCOMMA,   SIM  1773
     1               NLEAPEN, NRPAREN, NIBES                            SIM  1774
      NUM=4*SB(2)+2*SB(2)+SB(1)+1                                       SIM  1775
      IF(CODE.NE.12) GO TO 2                                            SIM  1776
      IF(CHAN.NE.21.AND.CHAN.NE.22.AND.CHAN.NE.23.AND.CHAN.NE.24)GO TO 2 SIM  1777
      IF(NUM.EQ.6) GO TO 25                                             SIM  1778
      WRITE(5,115)                                                      SIM  1779
      WRITE(6,115)                                                      SIM  1780
      KFLAG=1                                                           SIM  1781
      RETURN                                                            SIM  1782
    2 GO TO (1,5,10,15,20,25,35,40) NUM                                 SIM  1783
    1 IF(SIGNAL.EQ.1.AND.REGISTR.EQ.1) WRITE(6,100)                     SIM  1784
      RETURN                                                            SIM  1785
    5 CHAN=22                                                           SIM  1786
      GO TO 45                                                          SIM  1787
   10 WRITE(6,105)(A(25-I1),I1=1,24)                                    SIM  1788
      RETURN                                                            SIM  1789
   15 CHAN=21                                                           SIM  1790
      IF(SIGNAL.EQ.1.AND.REGISTR.EQ.1) WRITE(6,110)                     SIM  1791
      GO TO 45                                                          SIM  1792
   20 CHAN=24                                                           SIM  1793
      GO TO 45                                                          SIM  1794
   25 DO 30 I1=1,24                                                     SIM  1795
   30 L(I1)=A(I1)                                                       SIM  1796
      IF(REGISTR.EQ.3) RETURN                                           SIM  1797
      REGIST=NREG(2)                                                    SIM  1798
      CALL REG2                                                         SIM  1799
      RETURN                                                            SIM  1800
   35 CHAN=23                                                           SIM  1801
      GO TO 45                                                          SIM  1802
   40 CHAN=25                                                           SIM  1803
   45 CALL LOAD                                                         SIM  1804
      RETURN                                                            SIM  1805
  100 FORMAT(' FLAGGED INSTRUCTION STORE CODE - IDLE')                  SIM  1806
  105 FORMAT(' FLAGGED INSTRUCTION TELEMETRY SIGNAL - ',4(6O1,1X))      SIM  1807
  110 FORMAT(' FLAGGED INSTRUCTION STORE IN "NOT STORAGE" CHANNEL')     SIM  1808
  115 FORMAT(' FLAGSTORE IS ALLOWED ONLY IN L-REG WHEN STORE INSTRUCTION SIM  1809
     1 IS TO CHAN 21, F, M, OR E-LOOPS -PROGRAM TERMINATED')            SIM  1810
      END                                                               SIM  1811
      SUBROUTINE DISPLAY
      INTEGER A(24), CHAN, CODE, COMREG(24,2), DD, E(3), EN, F(6), FC
      INTEGER H(16), ON, CNE, PHASE, R(24,4), REGIST, REGISTR, RI, RK
      INTEGER SR(7), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,19), XI
      INTEGER Y(24,19), YI, ZERO, RES(10), DISPOSE
      COMMON A, CHAN, CODE, COMREG, CC, E, EN, F, FC, H, I(24), IRES
      COMMON L(24), M(128,21), N(24), NCOL, NFLAG, NUM, NWORD(72), PHASE
      COMMON SR, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK
      COMMON X, XI, Y, YI, DISPOSE
      COMMON/NNDATA/ ON, ZERO, CNE, NRES(10), NN(27), NBLANK, NCOMMA,
     1               NLPAREN, NRPAREN, NINES
      ENTRY REG2
      DO 1 I1=1,10
      IF(REGIST.EQ.NRES(I1)) GO TO 5
    1 CONTINUE
      RETURN
    5 IF(RES(I1).EQ.1) GO TO(10,15,20,25,30,40,50,60,70,75) I1
      RETURN
   10 WRITE(6,110)(A(25-I1),I1=1,24)
      RETURN
   15 WRITE(6,115)(L(25-I1),I1=1,24)
      RETURN
   20 WRITE(6,120)(I(25-I1),I1=1,24)
      RETURN
   25 WRITE(6,125)(N(25-I1),I1=1,24)
      RETURN
   30 I2=U
      DO 35 I1=1,24
      COMREG(I1)=AND(I2,CNE)
   35 I2=SHIFT(I2,-1)
      WRITE(6,135)(COMREG(25-I1),I1=1,24)
      RETURN
   40 I2=MOD(SECT-1,6)+1
      I3=F(I2)
      DO 45 I1=1,24
      COMREG(I1)=AND(I3,CNE)
   45 I3=SHIFT(I3,-1)
      I2=I2-1
      WRITE(6,135) I2,(COMREG(25-I1),I1=1,24)
      RETURN
   50 I2=MOD(SECT-1,3)+1
      I3=E(I2)
      DO 55 I1=1,24
      COMREG(I1)=AND(I3,CNE)
   55 I3=SHIFT(I3,-1)
      I2=I2-1
      WRITE(6,140) I2,(COMREG(25-I1),I1=1,24)
      RETURN
   60 I2=MOD(SECT-1,15)+1
      I3=H(I2)
      DO 65 I1=1,24
      COMREG(I1)=AND(I3,CNE)
   65 I3=SHIFT(I3,-1)
      I2=I2-1
      WRITE(6,145) I2,(COMREG(25-I1),I1=1,24)
      RETURN
   70 I2=MOD(SECT-1,4)+1
      I3=I2-1
      WRITE(6,150) I3,(V(25-I1,I2),I1=1,24)
      RETURN
   75 I2=MOD(SECT-1,4)+1
      I3=I2-1
      WRITE(6,155) I3,(R(25-I1,I2),I1=1,24)
      RETURN
      ENTRY REG1
      DO 80 I1=1,10
   80 RES(I1)=0
      REGISTR=0
      DO 85 I1=NCOL,72
      IF(NWORD(I1).EQ.NLPAREN) GO TO 90
   85 CONTINUE
      WRITE(6,[illegible])(NWORD(I1),I1=NCOL,72)
      WRITE(6,[illegible])(NWORD(I1),I1=NCOL,72)
      NCOL=I1
      RETURN
   90 NCOL=I1
   95 NCOL=NCOL+1
      IF(NCOL.GT.72) RETURN
      IF(NWORD(NCOL).EQ.NCOMMA.OR.NWORD(NCOL).EQ.NBLANK) GO TO 95
      IF(NWORD(NCOL).EQ.NRPAREN) RETURN
      DO 100 I1=1,10
      IF(NWORD(NCOL).EQ.NRES(I1)) GO TO 105
```

```
  100 CONTINUE                                                         SIM    1894
      WRITE(5,165) NWORD(NCOL)                                         SIM    1895
      WRITE(6,165) NWORD(NCOL)                                         SIM    1896
      GO TO 95                                                         SIM    1897
  105 REG(II)=1                                                        SIM    1898
      REGISTR=1                                                        SIM    1899
      GO TO 95                                                         SIM    1900
  110 FORMAT(* A(24-1) = *,8(3O1,1X))                                  SIM    1901
  115 FORMAT(* L(24-1) = *,3(3O1,1X))                                  SIM    1902
  120 FORMAT(* B(24-1) = *,8(3O1,1X))                                  SIM    1903
  125 FORMAT(* Q(24-1) = *,2(3O1,1X))                                  SIM    1904
  130 FORMAT(* U(24-1) = *,3(3O1,1X))                                  SIM    1905
  135 FORMAT(* F(*,I1,*,24-1) = *,8(3O1,1X))                           SIM    1906
  140 FORMAT(* E(*,I1,*,24-1) = *,8(3O1,1X))                           SIM    1907
  145 FORMAT(* M(*,I2,*,24-1) = *,3(3O1,1X))                           SIM    1908
  150 FORMAT(* Y(*,I1,*,24-1) = *,3(3O1,1X))                           SIM    1909
  155 FORMAT(* R(*,I1,*,24-1) = *,8(3O1,1X))                           SIM    1910
  160 FORMAT(* REGISTER DISPLAY REQUEST IS INVALID - *,72R1)           SIM    1911
  165 FORMAT(1H ,R1,* IS NOT A VALID REGISTER DISPLAY ARGUMENT*)       SIM    1912
      END                                                              SIM    1913
      SUBROUTINE REPORT
      INTEGER A(24), CHAN, CODE, CORREG(24,2), DD, E(9), EN, F(4), FC
      INTEGER H(10), CK, ONE, PHASE, R(24,4), REGIST, REGISTR, RI, RK
      INTEGER S2(3), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,13), XI
      INTEGER Y(24,19), YI, ZERO, DISPOSE
      COMMON A, CHAN, CODE, CORREG, DD, E, EN, F, FC, H, I(24), IREG
      COMMON L(24), M(128,21), N(24), NCOL, NFLAG, NUM, NWORD(72), PHASE
      COMMON S2, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK
      COMMON X, XI, Y, YI, DISPOSE
      COMMON/HDATA/ CK, ZERO, ONE, KREG(19), NH(27), NBLANK, NCOMMA,
     1              NLPAREN, NRPAREN, NINES
      WRITE(6,100)
      NUM=ZERO
    1 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 55
      IF(NWORD(NCOL).EQ.NLPAREN) GO TO 5
      IF(NWORD(NCOL).EQ.NBLANK) GO TO 55
      GO TO 1
    5 IF(NWORD(NCOL+1).EQ.NH(21)) GO TO 15
   10 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 20
      IF(NWORD(NCOL).EQ.NBLANK) GO TO 20
      IF(NWORD(NCOL).EQ.NRPAREN) GO TO 20
      GO TO 10
   15 NCOL=NCOL+1
      IF(NCOL.GT.72) GO TO 55
      IF(NWORD(NCOL).EQ.NBLANK) GO TO 55
      IF(NWORD(NCOL).EQ.NRPAREN) GO TO 55
      GO TO 15
   20 IF(DISPOSE.EQ.NH(18)) GO TO 80
      DO 50 I3=1,21
      CODE=ZERO
      DO 45 I2=1,128,2
      IF(M(I2,I3).EQ.NINES.AND.M(I2+1,I3).EQ.NINES) GO TO 45
      DO 40 I1=1,2
      PHASE=M(I1+I2-1,I3)
      IF(PHASE.EQ.NINES) GO TO 30
      DO 25 I4=1,24
      CORREG(I4,I1)=AND(PHASE,ONE)
   25 PHASE=SHIFT(PHASE,-1)
      GO TO 40
   30 DO 35 I4=1,24
   35 CORREG(I4,I1)=77777777B
   40 CONTINUE
      WRITE(6,105) NUM,CODE,((CORREG(25-I4,I1),I4=1,24),I1=1,2)
   45 CODE=CODE+1228
   50 NUM=NUM+328
      WRITE(6,110)
      RETURN
   55 IF(DISPOSE.EQ.NH(18)) GO TO 80
      DO 75 I3=1,21
      CODE=ZERO
      DO 70 I2=1,128,4
      IF(M(I2,I3).EQ.NINES.AND.M(I2+1,I3).EQ.NINES.AND.M(I2+2,I3).EQ.
     1   NINES.AND.M(I2+3,I3).EQ.NINES) GO TO 70
      DO 65 I1=1,4
      IF(M(I1+I2-1,I3).EQ.NINES) GO TO 60
      CORREG(I1)=M(I1+I2-1,I3)
      GO TO 65
   60 CORREG(I1)=77777777B
   65 CONTINUE
      WRITE(6,115) NUM,CODE,(CORREG(I1),I1=1,4)
   70 CODE=CODE+348
   75 NUM=NUM+328
      WRITE(6,110)
      RETURN
   80 DO 87 I3=1,21
      CODE=ZERO
      DO 85 I2=1,128,4
      IF(M(I2,I3).EQ.NINES.AND.M(I2+1,I3).EQ.NINES.AND.M(I2+2,I3).EQ.
     1   NINES.AND.M(I2+3,I3).EQ.NINES) GO TO 85
      DO 83 I1=1,4
      PHASE=M(I1+I2-1,I3)
      IF(PHASE.EQ.NINES) GO TO 82
      DO 81 I4=1,24
      CORREG(I4,I1)=AND(PHASE,ONE)
   81 PHASE=SHIFT(PHASE,-1)
      GO TO 84
   82 DO 83 I4=1,24
   83 CORREG(I4,I1)=77777777B
   84 CONTINUE
      WRITE(6,115) NUM,CODE,((CORREG(25-I4,I1),I4=1,24),I1=1,4)
```

```
55 CODE=CODE+242
57 NUM=NUM+323
   WRITE(6,119)
   RETURN
56 DO 97 I3=1,22
   CODE=7580
   DO 95 I2=1,125,8
   IF(M(I2,I3).EQ.MINES.AND.M(I2+1,I3).EQ.MINES.AND.M(I2+2,I3).EQ.
  1   MINES.AND.M(I2+3,I3).EQ.MINES.AND.M(I2+4,I3).EQ.MINES.AND.
  2   M(I2+5,I3).EQ.MINES.AND.M(I2+6,I3).EQ.MINES.AND.M(I2+7,I3)
  3   .EQ.MINES) GO TO 95
   DO 93 I1=1,8
   IF(M(I1+I2-1,I3).EQ.MINES) GO TO 91
   GO TO 93
91 M(I1+I2-1,I3)=77777777B
93 CONTINUE
   WRITE(6,115) NUM,CODE,(M(I1+I2-1,I3),I1=1,8)
95 CODE=CODE+10B
97 NUM=NUM+320
   WRITE(6,119)
   RETURN
100 FORMAT(/ /,"  ** MEMORY DUMP **",//," CHAN  SECT",/)
105 FORMAT(1H,,I4,O2,3X,O3,3X,2(8O1,1X,8O1,1X,8O1,1X,8O1,3X))
110 FORMAT(1H,,I4,O2,3X,O3,3X,4(8O1,1X,8O1,1X,8O1,1X,8O1,3X))
119 FORMAT(" ** END OF MEMORY DUMP **",5X,"(PORTIONS OF MEMORY NOT LIS
  1TED CONTAIN NO  INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN)"/
  2/)
115 FORMAT(1H,,I4,O2,3X,O3,4(7X,O8))
116 FORMAT(1H ,1X,O2,3X,O3,8(7X,O8))
   END
```

```
      SUBROUTINE DISCRET
      INTEGER A(24), CHAR, CODE, COMREG(24,2), DO, E(3), EN, F(4), FC
      INTEGER H(15), IN, ONE, PHASE, R(24,4), REGIST, REGISTR, RI, RK
      INTEGER SR(3), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,19), XI
      INTEGER Y(24,19), YI, ZERO, DISPOSE
      COMMON A, CHAR, CODE, COMREG, DO, E, EN, F, FC, H, I(24), IRES
      COMMON L(24), M(128,24), N(24), NCOL, LFLAG, NUP, INWORD(72), PHASE
      COMMON SR, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK
      COMMON X, XI, Y, YI, DISPOSE
      COMMON/INDATA/ IN, ZERO, ONE, KREG(13), NN(27), NBLANK, NCOMMA,
     1              NLPAREN, NRPAREN, NTRES
      ENTRY DISX
      XI=XI+1
      IF(XI.GT.19) GO TO 25
    1 NCOL=NCOL+1
      IF(NCOL.GT.72) RETURN
      IF(INWORD(NCOL).EQ.NLPAREN) GO TO 5
      GO TO 1
    5 DO 15 I1=1,19
      X(20-I1,XI)=ZERO
   10 NCOL=NCOL+1
      IF(INWORD(NCOL).EQ.NBLANK) GO TO 10
      IF(INWORD(NCOL).NE.NN(1).AND.INWORD(NCOL).NE.NN(2)) WRITE(6,100)
     1   INWORD(NCOL)
      IF(INWORD(NCOL).EQ.NN(2)) X(20-I1,XI)=ONE
   15 CONTINUE
   20 NCOL=NCOL+1
      IF(NCOL.GT.72) RETURN
      IF(INWORD(NCOL).EQ.NRPAREN) RETURN
      GO TO 20
   25 WRITE(6,105)
      WRITE(4,105)
      GO TO 20
      ENTRY DISY
      YI=YI+1
      IF(YI.GT.19) GO TO 50
   30 NCOL=NCOL+1
      IF(NCOL.GT.72) RETURN
      IF(INWORD(NCOL).EQ.NLPAREN) GO TO 35
      GO TO 30
   35 DO 45 I1=1,24
      Y(25-I1,YI)=ZERO
   40 NCOL=NCOL+1
      IF(INWORD(NCOL).EQ.NBLANK) GO TO 40
      IF(INWORD(NCOL).NE.NN(1).AND.INWORD(NCOL).NE.NN(2)) WRITE(6,110)
     1   INWORD(NCOL)
      IF(INWORD(NCOL).EQ.NN(2)) Y(25-I1,YI)=ONE
   45 CONTINUE
      GO TO 20
   50 WRITE(6,115)
      WRITE(4,115)
      GO TO 20
  100 FORMAT(" THE SYMBOL "",A1,"" IS NOT A VALID X-DISCRETE SIGNAL.  IT
     1 HAS BEEN REPLACED BY ZERO")
  105 FORMAT(" MAXIMUM ALLOWED X-DISCRETE SIGNAL REQUESTS IS 19 - PRESE
     1NT REQUEST NOT ACCEPTED")
  110 FORMAT(" THE SYMBOL "",A1,"" IS NOT A VALID Y-DISCRETE SIGNAL.  IT
     1 HAS BEEN REPLACED BY ZERO")
  115 FORMAT(" MAXIMUM ALLOWED Y-DISCRETE SIGNAL REQUESTS IS 19 - PRESE
     1NT REQUEST NOT ACCEPTED")
      END
      SUBROUTINE INCREME                                                SIM    2087
      INTEGER A(24), CHAN, CODE, COMREG(24,2), CO, E(8), EN, F(4), FC   SIM    2088
      INTEGER M(19), ON, ONE, PHASE, R(24,4), REGIST, REGISTR, RI, RK   SIM    2089
      INTEGER SB(3), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,19), XI     SIM    2090
      INTEGER Y(24,19), YI, ZERO, DISPOSE                               SIM    2091
      COMMON A, CHAN, CODE, COMREG, CO, E, EN, F, FC, M, I(24), IREG    SIM    2092
      COMMON L(24), M(128,21), K(24), NCOL, NFLAG, NEW, NWORD(72), PHASE SIM   2093
      COMMON SB, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK SIM    2094
      COMMON X, XI, Y, YI, DISPOSE                                      SIM    2095
      COMMON/NXDATA/ ON, ZERO, ONE, NRES(19), NX(27), NBLANK, NCOMMA,   SIM    2096
     1               NLPAREN, NRPAREN, SIDES                            SIM    2097
      ENTRY INCR                                                        SIM    2098
      RI=RI+1                                                           SIM    2099
      IF(RI.GT.4) RI=1                                                  SIM    2100
    1 NCOL=NCOL+1                                                       SIM    2101
      IF(NCOL.GT.72) RETURN                                             SIM    2102
      IF(NWORD(NCOL).EQ.NLPAREN) GO TO 5                                SIM    2103
      GO TO 1                                                           SIM    2104
    5 DO 15 I1=1,24                                                     SIM    2105
      R(25-I1,RI)=ZERO                                                  SIM    2106
   10 NCOL=NCOL+1                                                       SIM    2107
      IF(NCOL.GT.72) RETURN                                             SIM    2108
      IF(NWORD(NCOL).EQ.NRPAREN) GO TO 16                               SIM    2109
      IF(NWORD(NCOL).EQ.NBLANK) GO TO 15                                SIM    2110
      IF(NWORD(NCOL).NE.NX(1).AND.NWORD(NCOL).NE.NX(2)) WRITE(6,100)    SIM    2111
     1   NWORD(NCOL)                                                    SIM    2112
      IF(NWORD(NCOL).EQ.NX(2)) R(25-I1,RI)=ONE                          SIM    2113
   15 CONTINUE                                                          SIM    2114
   16 SECT=RI                                                           SIM    2115
      REGIST=NREG(10)                                                   SIM    2116
   20 IF(REGISTR.EQ.0) GO TO 25                                         SIM    2117
      CALL RES2                                                         SIM    2118
   25 NCOL=NCOL+1                                                       SIM    2119
      IF(NCOL.GT.72) RETURN                                             SIM    2120
      IF(NWORD(NCOL).EQ.NRPAREN) RETURN                                 SIM    2121
      GO TO 25                                                          SIM    2122
      ENTRY INCV                                                        SIM    2123
      VI=VI+1                                                           SIM    2124
      IF(VI.GT.4) VI=1                                                  SIM    2125
   30 NCOL=NCOL+1                                                       SIM    2126
      IF(NCOL.GT.72) RETURN                                             SIM    2127
      IF(NWORD(NCOL).EQ.NLPAREN) GO TO 35                               SIM    2128
      GO TO 30                                                          SIM    2129
   35 DO 45 I1=1,24                                                     SIM    2130
      V(25-I1,VI)=ZERO                                                  SIM    2131
   40 NCOL=NCOL+1                                                       SIM    2132
      IF(NCOL.GT.72) RETURN                                             SIM    2133
      IF(NWORD(NCOL).EQ.NRPAREN) GO TO 46                               SIM    2134
      IF(NWORD(NCOL).EQ.NBLANK) GO TO 45                                SIM    2135
      IF(NWORD(NCOL).NE.NX(1).AND.NWORD(NCOL).NE.NX(2)) WRITE(6,105)    SIM    2136
     1   NWORD(NCOL)                                                    SIM    2137
      IF(NWORD(NCOL).EQ.NX(2)) V(25-I1,VI)=ONE                          SIM    2138
   45 CONTINUE                                                          SIM    2139
   46 SECT=VI                                                           SIM    2140
      REGIST=NREG(9)                                                    SIM    2141
      GO TO 20                                                          SIM    2142
  100 FORMAT(* THE SYMBOL *,A1,* IS NOT A VALID R-INCREMENTAL INPUT.    SIM    2143
     1IT HAS BEEN REPLACED BY ZERO*)                                    SIM    2144
  105 FORMAT(* THE SYMBOL *,A1,* IS NOT A VALID V-INCREMENTAL INPUT.    SIM    2145
     1IT HAS BEEN REPLACED BY ZERO*)                                    SIM    2146
      END                                                               SIM    2147
```

Appendix B

D17B Instruction Set

and

D17B Load Codes

D17B Instruction Set

CODE	DESCRIPTION	NUMERIC CODE	WORD TIMES
ADD	Add	~~64 c,s~~	1
ALS	Accumulator Left Shift	00 22,s	s+1
ANA	Logical And to Accumulator	40 42,s	1
ARS	Accumulator Right Shift	00 32,s	s+1
BOA	Binary Output A	40 10,s	1
BOB	Binary Output B	40 12,s	1
BOC	Binary Output C	40 02,s	1
CLA	Clear and Add to Accumulator	44 c,s	1
COA	Character Output A	00 40,s	s+1
COM	Complement	40 46,s	1
DIA	Discrete Input A	~~40 52,s~~	1
DIB	Discrete Input B	40 50,s	1
DOA	Discrete Output A	40 26,s	1
EFC	Enter Fine Countdown	40 62,s	1
HFC	Halt Fine Countdown	40 60,s	1
HPR	Halt and Proceed	40 22,s	1
LPR	Load Phase Register	40 7-,s	1
MIM	Minus Magnitude	40 44,s	1
MPM	Multiply Modified	34 c,s	13
MPY	Multiply	24 c,s	13
RSD	Reset Detector	40 20,s	1
SAD	Split Add	60 c,s	1
SAL	Split Accumulator Left Shift	00 20,s	s+1
SAR	Split Accumulator Right Shift	00 30,s	s+1
SCL	Split Compare and Limit	04 c,s	2
SLL	Split Left Word Left Shift	00 24,s	s+1
SLR	Split Left Word Right Shift	00 34,s	s+1
SMM	Split Multiply Modified	30 c,s	7
SMP	Split Multiply	20 c,s	7
SRL	Split Right Word Left Shift	00 26,s	s+1
SRR	Split Right Word Right Shift	00 36,s	s+1
SSU	Split Subtract	70 c,s	1
STO	Store Accumulator	54 c,s	1

D17B Instruction Set (cont'd)

CODE	DESCRIPTION	NUMERIC CODE	WORD TIMES
SUB	Subtract	~~74 c,s~~	1
TMI	Transfer on Minus	10 c,s	1
TRA	Transfer	50 c,s	1
VOA	Voltage Output A	40 30,s	1
VOB	Voltage Output B	40 32,s	1
VOC	Voltage Output C	40 34,s	1

D17B Load Codes

HALT	- This load code causes the D17B to stop accepting data and enter the program halt mode. The manual halt mode will be entered if the compute switch is set at the Halt position.
COMPUTE	- This code causes the computer to go to the manual halt mode of noncompute. The compute mode will be entered if the compute switch is set at the Run or Single position.
FILL	- This load code "0" sets the fill/verify flipflop (O_3).
VERIFY	- This load code "1" sets the fill/verify flipflop (O_3).
LOCATE	- This code causes the contents of the Lower Accumulator to be shifted into the Instruction Register.
CLEAR	- This code clears the Lower Accumulator by filling it with all zeros.
DELETE	- This code causes no action.
ENTER	- The action produced by this code depends upon the setting of the fill/verify flipflop. When this code is first deciphered, the contents of the Lower Accumulator is transferred into the Accumulator.

If O_3 is "0" set, then the contents of the Accumulator is stored in the memory location addressed by the Instruction Register.

If O_3 is "1" set, then the contents of the Accumulator is compared with the contents of the memory location addressed by the Instruction Register.

The last action of this load code is to increment the Instruction Register by one.

Appendix C

Figures for Interpreting Binary, Discrete, and Voltage Outputs

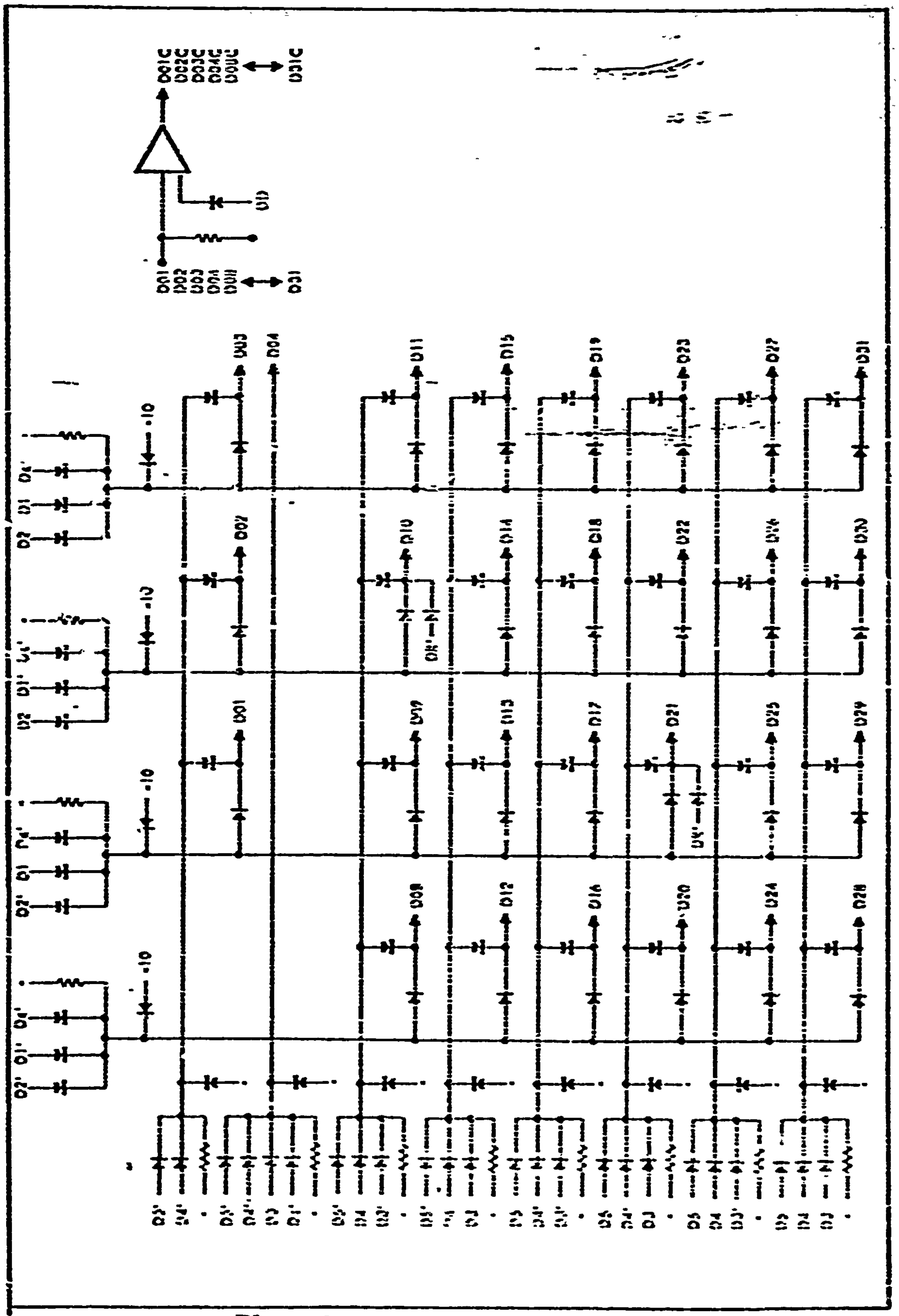


Fig. 12. Discrete Outputs (Ref 4:TR-8)

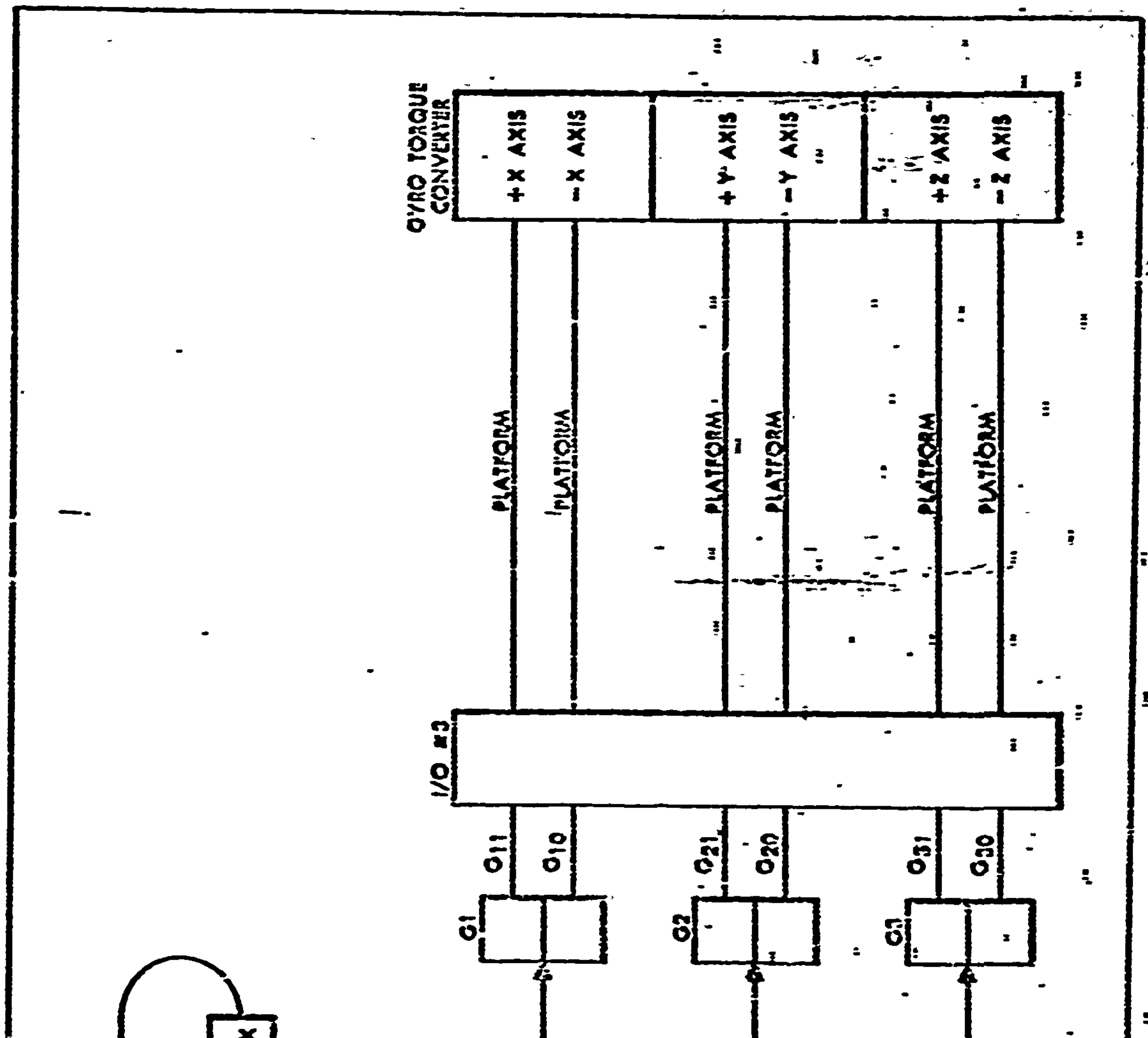


Fig. 13. Binary Outputs (Ref 5:57)

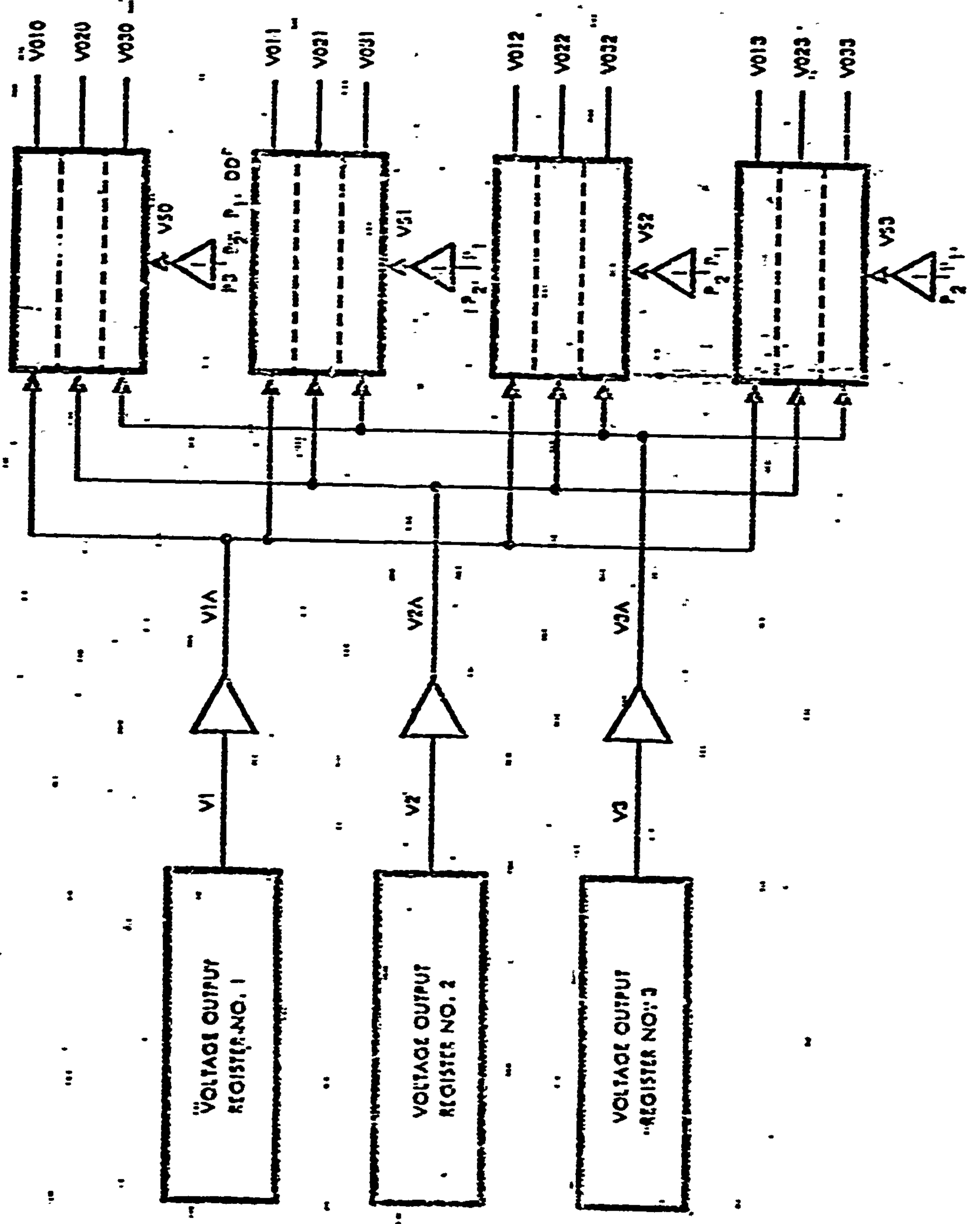

V010
V020
V030
V011
V021
V031
V012
V022
V032
V013
V023
V033
VS0
VS1
VS2
VS3
V1A
V2A
V3A
V1
V2
V3
VOLTAGE OUTPUT REGISTER NO. 1
VOLTAGE OUTPUT REGISTER NO. 2
VOLTAGE OUTPUT REGISTER NO. 3

Fig. 14. Voltage Output Scheme (Ref 5:56)

Appendix D

Instructions for Using
the D17B Computer Simulation Program
at AFIT

Foreword to Appendix D

A software simulation program has been written which simulates the functions of the Minuteman D17B computer. The structure and organization of this simulation program is described in chapter II of this thesis. The D17B simulation language is presented in chapter III, chapter IV contains a listing of the error statements of the simulation program, and chapter V is made up of example programs which were run on the simulated computer.

This appendix contains information for using the D17B computer simulation program at AFIT. Procedures are given for accessing the simulation program from a teletype terminal. Information concerning the use of program tapes and external program files is also included. A condensed version of the D17B simulation language is given followed by the listing of a method for creating a shortened version of the simulation language.

Procedures for Using the D17B Computer Simulation Program at AFIT

The D17B computer simulation program was written to be used from a teletype terminal. Procedures for operating a teletype terminal are contained in the Intercom 2 Reference Manual (Ref 3).

The simulation program is available on the CDC 6600 Computer System as a permanent file. HHSIMULATION is the permanent file name. Only one version of the program exists so a cycle number need not be specified when attaching the permanent file. However, the program was catalogued as CY=1. The simulation program requires less than 40K of memory to execute on the 6600 system. The majority of programs run on the simulation program require less than 5 seconds of central processor time.

Operation from a Teletype Terminal. To access the simulation program from the teletype terminal requires the user to LOGIN with the 6600 computer. Procedures for doing this are contained in the Intercom 2 Reference Manual (Ref 3: Chap. 3, p. 3). After login has been successfully completed the teletype prints out

```
    COMMAND-
```

The user will respond with

```
    ATTACH,HMAN,HHSIMULATION,MR=1.
```

to make the simulation program available for his use. The system will respond to this request by typing the time and the attach request followed by

```
COMMAND-
```

The user must decide if he wants the output to be printed at the teletype or if he wants to dispose the output to the batch terminal line printer. If output is to be printed by the teletype the user should respond to the command with

```
CONNECT,INPUT,OUTPUT.
```

If output is to be disposed to the high speed printer the user should respond with

```
CONNECT,INPUT,TAPE4.
```

The system will process this request and the teletype will print the following:

```
COMMAND-
```

The user should respond with

```
MHAM.
```

which puts the simulation program into execution. During execution of his data the user should respond to any messages that appear at the teletype. When the user has finished running programs he should exit the simulation program by specifying FR(OFF) and respond to the message:

```
     "TO RUN ANOTHER PROGRAM TYPE 'RUN'; TO STOP TYPE
'HALT' -
```

by typing

```
HALT
```

The system responds with the message

```
"END OF PROGRAM"
```

and prints the amount of execution time used. This will be followed by

```
COMMAND-
```

If output is to be disposed to the batch terminal line printer, the user should respond with

```
DISPOSE,OUTPUT,P2=ET7.
```

followed by the LOGOUT procedures contained in the Intercom 2 Reference Manual (Ref 3:Chap. 3, p. 3-6).

If output is not to be ~~disposed to the line~~ printer, then the user should follow the procedures for LOGOUT.

<u>Using Program Tapes</u>. Program tapes can supply data to the simulation program. The data on these tapes is entered by the tape reader located at each teletype terminal. The program tapes can also be punched at the teletype terminal. Procedures for punching and reading of program tapes is contained in the Intercom 2 Reference Manual (Ref 3:Chap. 2, p. 7-9).

Program tapes which will also be run on the D17B computer must be written using the ASCII representation described in chapter III of this thesis. Blanks, line feeds, and carriage returns are ignored by the D17B computer tape reader, so these symbols can be punched on the tapes. This allows the program tapes which will be run on the D17B computer to also be executed by the simulation program.

Using External Files. Two files can be established in the CDC 6600 computer which will supply data in the ASCII representation to the simulation program. The files must have the names of TAPE2 or TAPE3. They can be created at the teletype by entering SETUP when the teletype prints

```
    COMMAND-
```

The user responds with

```
    SETUP.
```

The teletype will process this command and print back

```
    NEW OR OLD FILE--
```

The user should type

```
    NEW/TAPE2   or   NEW/TAPE3
```

The teletype will respond with

```
    READY.
```

The user can proceed to write a program in ASCII representation. Each line of the program must be proceeded by a line number. Procedures for creating programs in SETUP are contained in the INTERCOM 2 Reference Manual (Ref 3:Chap. 4). The last symbol supplied must be the letter "N". This symbol signifies the end of the program. Also the SAVE directive should always be given at the end of a program.

To use the program created on TAPE2 or TAPE3, the user must provide an argument to the simulation program command NMAN. NMAN(2) will result in the reading of the data on TAPE2. NMAN(3) causes the simulation program to read the data from TAPE3. An example in which data is read from TAPE2 is as follows:

```
(ENTER PROGRAM)

PR(ON) MR(ON) EN(ON) PS(ON) HHAN(2)
MR(ON) K(RUN)
PR(OFF)
```

D17B Computer Simulation Program Language. A listing of the simulation language is as follows:

Octal numbers - 0, 1, 2, 3, 4, 5, 6, 7

Binary numbers - 0, 1

Load Codes - HALT, LOCATION, FILL, VERIFY
COMPUTE, ENTER, CLEAR, DELETE

When OCTAL is specified, input must be in Octal representation. When BINARY is specified, input must be in Binary representation. When HHAN is specified, input must be in ASCII representation. (Default is OCTAL)

		Octal Representation	Binary Representation	ASCII Representation
Numbers -		0	10000	0
		1	00001	1
		2	00010	2
		3	10011	3
		4	00100	4
		5	10101	5
		6	10110	6
		7	00111	7
Load Codes -		HALT	01000	8
		LOCATION	11001	9
		FILL	11010	Z
		VERIFY	01011	;
		COMPUTE	11100	<
		ENTER	01101	=

CLEAR	01110	^
DELETE	11111	?

	Switch Name	Mnemonic(Setting)
Switches -	Timing Signal	T(ON)
	Power On/Off Switch	PR(ON), PR(OFF)
	Initiate Load Switch	FS(ON)
	Master Reset Switch	MR(ON)
	Cold-Storage Write Switch	EW(ON), EW(OFF)
	Discrete Switch	DD(ON), DD(OFF)
	Mechanical Input Switch	IM(ON)

(Default of these switches is OFF)

Compute Mode Switch	K(HALT), K(SINGLE), K(RUN)

(Default of this switches is HALT)

Display - A binary output listing of any of the following registers or loops will be given whenever its contents changes, if it appears as the argument of the REGISTER command:

Mnemonic	Register or Loop
A	Accumulator
I	Instruction Register
L	Lower Accumulator
N	Number Register
F	F-loop
E	E-loop
H	H-loop
U	U-loop
V	V-loop
R	R-loop

Example: REGISTER(A,I,L,N)
REGISTER(AILNEFUHVR)

(No. of arguments can vary from 0 to 10 and must be one of those given above)

The contents of memory will be given as output whenever a MEMORY command is given:

MEMORY(OCTAL)	Memory dump will be given in Octal.
MEMORY(BINARY)	Memory dump will be given in Binary.

(Default is OCTAL)

Discrete Inputs - X(19 bits as argument)
Y(24 bits as argument)

Example: X(1 000 000 101 111 110)
Y(111000111000111000111000)

Incremental Inputs - V(24 bits as argument)
R(24 bits as argument)

Example: V(0000 0001 0010 0011 0100 0101)
R(00001111 11110000 10101010)

Miscellaneous -

SIGNAL - Each time SIGNAL is used, it flips the representation from 0 to 1 or 1 to 0. SIGNAL is 0 at the beginning of the program run. When SIGNAL is 1, the modes of the computer will be traced.

EXECUTE(xxxx) - No. of execution cycles in the compute mode can be specified.

Example:

EXECUTE(0010)	10 executions
EXECUTE(9999)	9999 executions
EXECUTE(0250)	250 executions

(Default is 50 executions)

DR - DR flipflop is "1" set.

VK(0) - VK flipflop is "0" set.
VK(1) - VK flipflop is "1" set.

RK(0) - RK flipflop is "0" set.
RK(1) - RK flipflop is "1" set.

REINITIALIZE - Memory is initialized, binary output flipflops are set to +1, and discrete input counters are set to zero.

Shortened Version of Simulation Language. The following listing contains the simulation language words which can be shortened, the interpreting letters, ~~and an example~~ of a shortened version:

Simulation Language words which can be Shortened	Interpreting Letters	Example of a Shortened Version
HALT	H	HALT
LOCATION	L	LOC
FILL	FI	FILL
VERIFY	V	VER
COMPUTE	CO	COM
ENTER	EN	EN
CLEAR	CL	CL
DELETE	DE	DEL
OCTAL	O	OCT
BINARY	B	BIN
KMAN	KM	KMAN
SINGLE	S	SING
RUN	R	RUN
REGISTER(Arg)	RE(Arg)	REG(Arg)
MEMORY(BINARY)	ME(B)	MEM(B)
MEMORY(OCTAL)	ME(O)	MEM(O)
SIGNAL	S	SIG
EXECUTE(Arg)	EX(Arg)	EXEC(Arg)
REINITIALIZATION	REI	REINIT